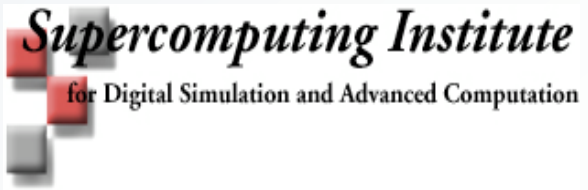
Introduction to UNIX Scripting with PERL

Cal Kirchhof and Yuk Sham

MSI Consultants

Phone: (612) 626 0802 (help)

Email: help@msi.umn.edu



Outline

- What is PERL?
- Why would I use PERL instead of something else?
- PERL features
  - How to run PERL scripts
  - PERL syntax, variables, quotes
  - Flow control constructs
  - Subroutines
- Typical UNIX scripting tasks
  - File filtering - matching & substitutions
  - Counting
  - Naming files
  - Executing applications & status checking
  - Mail
- More information

Supercomputing Institute
for Digital Simulation and Advanced Computation

What is PERL?

- Practical Extraction Report Language
  - Written by Larry Wall who also called it the "Pathologically Eclectic Rubbish Lister"
- Combines capabilities of Bourne shell, csh, awk, sed, grep, sort and C
- To assist with common tasks that are too heavy or portable-sensitive in shell, and yet too weird or too complicated to code in C or other programming language.
- File or list processing - matching, extraction, formatting (text reports, HTML, mail, etc.)

Supercomputing Institute
for Digital Simulation and Advanced Computation

Why would I use PERL instead of something else?

- Interpreted language
- Commonly used for cgi programs
- Very flexible
- Very automatic
- Can be very simple for a variety of tasks
- WIDELY available
- HIGHLY portable

Supercomputing Institute
for Digital Simulation and Advanced Computation

PERL features

- C-style flow control (similar)
- Dynamic allocation
- Automatic allocation
- Numbers
- Lists
- Strings
- Arrays
- Associative arrays (hashes)

Supercomputing Institute
for Digital Simulation and Advanced Computation

PERL features

- Very large set of publicly available libraries for wide range of applications
- Math functions (trig, complex)
- Automatic conversions as needed
- Pattern matching
- Standard I/O
- Process control
- System calls
- Can be object oriented

Supercomputing Institute
for Digital Simulation and Advanced Computation

How to run PERL scripts

```
% cat hello.pl
print "Hello world from PERL.\n";
%
```

```
% perl hello.pl
```

Hello world from PERL.

Supercomputing Institute
for Digital Simulation and Advanced Computation

How to run PERL scripts

OR ----------

```
% which perl
/usr/bin/perl

% cat hello.pl
#!/usr/bin/perl
print "Hello world from PERL.\n";


%chmod a+rx hello.pl
% hello.pl
```

Hello world from PERL.

(the .pl suffix is just a convention - no special meaning - to perl)
/usr/local/bin/perl is another place perl might be
linked at Institute

Supercomputing Institute
for Digital Simulation and Advanced Computation

PERL syntax

- Free form - whitespace and newlines are ignored, except as delimiters
- PERL statements may be continued across line boundaries
- All PERL statement end with a ; (semicolon)
- Comments begin with the # (pound sign) and end at a newline
  - no continuation
  - may be anywhere, not just beginning of line
- Comments may be embedded in a statement
  - see previous item

Supercomputing Institute
for Digital Simulation and Advanced Computation

Example 1:

Hello world

```
#!/usr/bin/perl
# This is how perl says hello
print "Hello world from PERL.\n";       # It says hello once
print "Hello world again from PERL.\n";# It says hello twice
```

Example 2:

```
#!/usr/bin/perl
print"Hello world from PERL.\n";print"Hello world again from PERL.\n";
```

Example 3:

```
#!/usr/bin/perl
print       "Hello world from PERL.\n";
print    "Hello world again from PERL.\n";
```

Hello world from PERL.
Hello world again from PERL.

Supercomputing Institute
for Digital Simulation and Advanced Computation

PERL variables

- Number or string
  - $count
- Array
  - List of numbers and/or strings
  - Indexed by number starting at zero
  - @an_array
- Associative array or hash
  - List of numbers and/or strings
  - Indexed by anything
  - %a_hash

Supercomputing Institute
for Digital Simulation and Advanced Computation

Strings and arrays

```
$x = 27;
$y = 35;
$name = "john";
@a = ($x,$y,$name);
print "x = $x and y = $y\n";
print "The array is @a \n";
```

```
X = 27 and y = 35
The array is 27 35 john
```

```
@a = ("fred","barney","betty","wilma");
print "The names are @a \n";
print "The first name is $a[0] \n";
print "The last  name is $a[3] \n";
```

```
The names are fred barney betty wilma
The first name is fred
The last name is wilma
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Associative arrays

```
$a{dad} = "fred";
$a{mom} = "wilma";
$a{child} = "pebble";
print "The mom is $a{mom} \n";
```

The mom is wilma

```
@keys = keys(%a);
@values = values(%a);
print "The keys are @keys \n"
print "The values are @values \n";
```

The keys are mom dad child

The values are wilma fred pebble

Supercomputing Institute
for Digital Simulation and Advanced Computation

Operators and functions

- increase or decrease existing value by 1 (++, --)
- modify existing value by +, -, * or /
  by an assigned value (+=, -=, *=, /=)

Example 1

```
$a = 1;
$b = "a";
++$a;
++$b;
print "$a $b \n";
```

```
2 b
```

Example 2

```
$a = $b = $c = 1;
++$b;
$c *= 3;
print "$a $b $c\n";
```

```
1 2 3
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Operators and functions

- Numeric logical operators

  ==, !=, <, >, <=, >=

- String logical operators

  eq, ne, lt, gt, le, ge

Supercomputing Institute
for Digital Simulation and Advanced Computation

Operators and functions

- Add and remove element from existing array (Push, pop, unshift, shift)
- Rearranging arrays (reverse, sort)

```
@a = qw(one two three four five six);
print "@a\n";
```

```
one two three four five six
```

```
unshift(@a,"zero");          # add elements to the array
print "@a\n";                # from the left side
```

```
zero one two three four five six
```

```
shift(@a);                   # removes elements from the array
print "@a\n";                # from the left side
```

```
one two three four five six
```

```
@a = reverse(@a);            # reverse the order of the array
print "@a\n";
```

```
six five four three two one
```

```
@a = sort(@a);               # sort the array in alphabetical order
print "@a\n";
```

```
five four one six three two
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Operators and functions

- Removes last character from a string (chop)
- Removes newline character, \n,from end of a string (chomp)
- Breaks a regular expression into fields (split) and joints the pieces back (join)

```
$a = "this is my expression\n";
print "$a";
```

this is my expression

```
chomp($a);
print "$a …. ";
@a = split(/ /,$a);                    # splits $a string into an array called @a
print "$a[3] $a[2] $a[1] $a[0]\n";
```

this is my expression…. expression my is this

```
$a = join(":",@_);                     # create a string called $a by joining
print "$a \n";                         # all the elements in the array @a and
                                       # having ":" spaced between them
```

this:is:my:expression

Supercomputing Institute
for Digital Simulation and Advanced Computation

Operators and functions

- Substituting a pattern (=~ s/..../...../)
- Transliteration (=~ tr/..../..../)

```
$_ = "this is my expression\n";
print "$_\n";
```

this is my expression

```
$_ =~ s/my/your/;
print "$_\n";
```

this is your expression

```
$_ =~ tr/a-z/A-Z/;
print "$_\n";
```

THIS IS YOUR EXPRESSION

Supercomputing Institute
for Digital Simulation and Advanced Computation

Flow control constructs

```
Control_operator  (expression(s) ) {
        statement_block;
}

Example:

if ( $i < $N ) {
        statement_block;
} else {
        statement_block;
}

foreach $i ( @list_of_items ) {
        statement_block;
}
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Subroutines

```
@a = qw(1 2 3 4);                  # assigns an array "@a"
print summation(@a),"\n";          # prints results of subroutine
                                   # summation using "@a" as
                                   # input


sub summation {                    # summing every element in
    my $k = 0;                     # the array "@a" and return
    foreach $i (@_) {              # the value as $k
        $k += $i;
    }
    return($k);
}
```

10

Supercomputing Institute
for Digital Simulation and Advanced Computation

Concatenating Strings with the . operator

```
$firstname = "George";
$midname =  "washington";
$lastname = "Bush";

$fullname = $lastname . ",  " . $firstname . " "
          . uc(substr $midname, 0, 1) . ".\n";

print $fullname;
```

Bush, George W.

Supercomputing Institute
for Digital Simulation and Advanced Computation

Sorting arrays and formatted output

```
@winners = ( ["Gandhi", 1982], ["Amadeus", 1984], ["Platoon", 1986],
          ["Rain Man", 1988], ["Braveheart", 1995], ["Titanic", 1997] );

@sortwinners = sort { $a->[0] cmp $b->[0] } @winners;

format STDOUT =
  @>>>>>>>>>>  @<<<<<
  $i->[0]     $i->[1]
.
foreach $i (@sortwinners) {
    write STDOUT;
}

print "\n(The list has " . scalar(@sortwinners) . " entries.)\n";
```

```
    Amadeus        1984
Braveheart         1995
     Gandhi        1982
    Platoon        1986
  Rain Man         1988
    Titanic        1997

(The list has 6 entries.)
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Command-line arguments

```
#!/usr/bin/perl

print "Command name: $0\n";
print "Number of arguments: $#ARGV\n";

for ($i=0; $i <= $#ARGV; $i++) {
  print "Arg $i is $ARGV[$i]\n";
}
```

% ./arguments.pl zero one two three

Number of arguments: 3
Arg 0 is zero
Arg 1 is one
Arg 2 is two
Arg 3 is three

Supercomputing Institute
for Digital Simulation and Advanced Computation

UNIX Environment Variables

```
print " your username is $ENV{'USER'} and \n";
print " your machine name is $ENV{'HOST'} and \n";
print " your display is set to $ENV{'DISPLAY'} and \n";
print " your shell is $ENV{'SHELL'} and \n";
print " your timezone is $ENV{'TZ'} etcetera.\n";
```

your username is shamy and
your machine name is cirrus.msi.umn.edu and
your display is set to localhost:10.0 and
your shell is /bin/tcsh and
your timezone is CST6CDT, etcetera...

Supercomputing Institute
for Digital Simulation and Advanced Computation

Typical UNIX scripting tasks

- Filter a file or a group of files
- Searching/Matching
- Naming file sequences
- Executing applications & status checking
- Counting files, lines, strings, etc.
- Report generation

Supercomputing Institute
for Digital Simulation and Advanced Computation

Filtering standard input

```
#!/usr/bin/perl
while( <> ) {                    # read from stdin one line at a time
    print "line $. : $_" ;       # print current line to stdout
}
```

print.txt

```
Silicon Graphics' Info Search lets you find all the information
available on a topic using a keyword search. Info Search looks
begin
through all the release notes, man pages, and/online books you
done
have installed on your system or on a networked server. From
the Toolchest on your desktop, choose Help-Info Search.
begin

Quick Answers tells you how to connect to an Internet Service Provider (ISP).
done
From the Toolchest on your desktop, choose
Help > Quick Answers > How Do I > Connect to an Internet Service Provider.
through all the release notes, man pages, and/online books you
Quick Answers tells you how to connect to an Internet Service Provider (ISP).
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Filtering standard input

./printlines.pl print.txt

```
line 1 : Silicon Graphics' Info Search lets you find all the information
line 2 : available on a topic using a keyword search. Info Search looks
line 3 : begin
line 4 : through all the release notes, man pages, and/online books you
line 5 : done
line 6 : have installed on your system or on a networked server. From
line 7 : the Toolchest on your desktop, choose Help-Info Search.
line 8 : begin
line 9 :
line 10 : Quick Answers tells you how to connect to an Internet Service Provider (ISP).
line 11 : done
line 12 : From the Toolchest on your desktop, choose
line 13 : Help > Quick Answers > How Do I > Connect to an Internet Service Provider.
line 14 : through all the release notes, man pages, and/online books you
line 15 : Quick Answers tells you how to connect to an Internet Service Provider (ISP).
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Filtering standard input

```
#!/usr/bin/perl
while( <> ) {
    print "line $. : $_" unless $. %2;        # print only the even lines
}
```

./printeven.pl print.txt

line 2 : available on a topic using a keyword search. Info Search looks
line 4 : through all the release notes, man pages, and/online books you
line 6 : have installed on your system or on a networked server. From
line 8 : begin
line 10 : Quick Answers tells you how to connect to an Internet Service Provider (ISP).
line 12 : From the Toolchest on your desktop, choose
line 14 : through all the release notes, man pages, and/online books you

Supercomputing Institute
for Digital Simulation and Advanced Computation

Filtering standard input

```
#!/usr/bin/perl
while( <> ) {
    if( /begin/ .. /done/ ) {
        print "line $. : $_";        # prints any text that
    }                                # starts with "begin"
}                                    # and finishes with "end"
```

./printpattern.pl print.text

```
line 3 : begin
line 4 : through all the release notes, man pages, and/online books you
line 5 : done
line 8 : begin
line 9 :
line 10 : Quick Answers tells you how to connect to an Internet Service Provider (ISP).
line 11 : done
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Filtering standard input

```
#!/usr/bin/perl
while( <> ) {
    if( /begin/ .. /done/ ) {
        unless( /begin/ || /done/ ) {
            print "line $. : $_";
        }
    }
}
```

./printpattern2.pl print.text

line 4 : through all the release notes, man pages, and/online books you
line 9 :
line 10 : Quick Answers tells you how to connect to an Internet Service Provider (ISP).

Supercomputing Institute
for Digital Simulation and Advanced Computation

sed Example

```
#!/usr/bin/perl
# sed.pl

my $expression = shift or "";

while( <> ) {
        $_ =~ eval $expression;
        print $_;
}
```

sed.txt

```
1:  Silicon Graphics' Info Search lets you find all the information
2:  available on a topic using a keyword search. Info Search looks
3:  through all the release notes, man pages, and/online books you
4:  have installed on your system or on a networked server. From
5:  the Toolchest on your desktop, choose Help-Info Search.
6:
7:  Quick Answers tells you how to connect to an Internet Service Provider (ISP).
8:  From the Toolchest on your desktop, choose
9:  Help > Quick Answers > How Do I > Connect to an Internet Service Provider.
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

sed

./sed.pl s/\[aeiou\]/_/gi sed.txt

```
1: S_l_c_n Gr_ph_cs' _nf_ S__rch l_ts y__ f_nd _ll th_ _nf_rm_t__n
2: _v__l_bl_ _n _ t_p_c _s_ng _ k_yw_rd s__rch. _nf_ S__rch l__ks
3: thr__gh _ll th_ r_l__s_ n_t_s, m_n p_g_s, _nd/_nl_n_ b__ks y__
4: h_v_ _nst_ll_d _n y__r syst_m _r _n _ n_tw_rk_d s_rv_r. Fr_m
5: th_ T__lch_st _n y__r d_skt_p, ch__s_ H_lp-_nf_ S__rch.
6:
7: Q__ck _nsw_rs t_lls y__ h_w t_ c_nn_ct t_ _n _nt_rn_t S_rv_c_ Pr_v_d_r
   (_SP).
8: Fr_m th_ T__lch_st _n y__r d_skt_p, ch__s_
9: H_lp > Q__ck _nsw_rs > H_w D_ _ > C_nn_ct t_ _n _nt_rn_t S_rv_c_ Pr_v_d_r.
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Naming files

- Files
- Reformating files

Supercomputing Institute
for Digital Simulation and Advanced Computation

Files

```
%cat mkfiles.pl
#!/usr/bin/perl
# touch.pl

foreach $i ( 0 .. 50 ) {
    print "touch gifdir/$i.gif\n";
    system("touch gifdir/$i.gif");
}
```

./touch.pl

Perl executes the following in unix:

```
touch gifdir/0.gif
touch gifdir/1.gif
touch gifdir/2.gif
touch gifdir/3.gif
touch gifdir/4.gif
    .
    .
    .
touch gifdir/48.gif
touch gifdir/49.gif
touch gifdir/50.gif
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Files

% ls –lt gifdir/*.gif

```
-rw-------   1 shamy   support   995343 Oct 21 18:50 50.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 49.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 48.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 47.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 46.gif
                                    .
                                    .
                                    .
-rw-------   1 shamy   support   995343 Oct 21 18:50 4.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 3.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 2.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 1.gif
-rw-------   1 shamy   support   995343 Oct 21 18:50 0.gif
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Files

```
#!/usr/bin/perl

foreach $i ( 0 .. 50 ) {
    $new = sprintf("step%3.3d.gif", $i);          # naming the gif file with
    print  "mv gifdir2/$i.gif gifdir2/$new\n";     # with a 3 digit numbering
    system "mv gifdir2/$i.gif gifdir2/$new";       # scheme
}
```

./rename.pl

Perl executes the following in unix:

```
mv  gifdir2/0.gif   gifdir2/step000.gif
mv  gifdir2/1.gif   gifdir2/step001.gif
mv  gifdir2/2.gif   gifdir2/step002.gif
mv  gifdir2/3.gif   gifdir2/step003.gif
mv  gifdir2/4.gif   gifdir2/step004.gif
          .
          .
mv  gifdir2/47.gif   gifdir2/step047.gif
mv  gifdir2/48.gif   gifdir2/step048.gif
mv  gifdir2/49.gif   gifdir2/step049.gif
mv  gifdir2/50.gif   gifdir2/step050.gif
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

ls gifdir2 (before)

Files

gifdir2:

0.gif 14.gif 2.gif 25.gif 30.gif 36.gif 41.gif 47.gif 7.gif
1.gif 15.gif 20.gif 26.gif 31.gif 37.gif 42.gif 48.gif 8.gif
10.gif 16.gif 21.gif 27.gif 32.gif 38.gif 43.gif 49.gif 9.gif
11.gif 17.gif 22.gif 28.gif 33.gif 39.gif 44.gif 5.gif
12.gif 18.gif 23.gif 29.gif 34.gif 4.gif 45.gif 50.gif
13.gif 19.gif 24.gif 3.gif 35.gif 40.gif 46.gif 6.gif

ls gifdir2 (after)

gifdir2:

script step008.gif step017.gif step026.gif step035.gif step044.gif
step000.gif step009.gif step018.gif step027.gif step036.gif step045.gif
step001.gif step010.gif step019.gif step028.gif step037.gif step046.gif
step002.gif step011.gif step020.gif step029.gif step038.gif step047.gif
step003.gif step012.gif step021.gif step030.gif step039.gif step048.gif
step004.gif step013.gif step022.gif step031.gif step040.gif step049.gif
step005.gif step014.gif step023.gif step032.gif step041.gif step050.gif
step006.gif step015.gif step024.gif step033.gif step042.gif
step007.gif step016.gif step025.gif step034.gif step043.gif

Supercomputing Institute
for Digital Simulation and Advanced Computation

Parsing and reformating Files

```
HEADER    CALCIUM-BINDING PROTEIN                29-SEP-92            1CLL   2
COMPND    CALMODULIN (VERTEBRATE)                                            1CLL   3
REMARK   1 REFERENCE 1                                                       1CLL  13
REMARK   1  AUTH   W.E.MEADOR,A.R.MEANS,F.A.QUIOCHO                          1CLL  14
RORIGX2     0.000000  0.018659  0.001155      0.00000                        1CLL 143
                    .
                    .
                    .
ATOM      1  N    LEU     4      -6.873  21.082  25.312    1.00 49.53        1CLL 148
ATOM      2  CA   LEU     4      -6.696  22.003  26.447    1.00 48.82        1CLL 149
ATOM      3  C    LEU     4      -6.318  23.391  25.929    1.00 46.50        1CLL 150
ATOM      4  O    LEU     4      -5.313  23.981  26.352    1.00 45.72        1CLL 151
ATOM      5  N    THR     5      -7.147  23.871  25.013    1.00 46.77        1CLL 152
ATOM      6  CA   THR     5      -6.891  25.193  24.428    1.00 46.84        1CLL 153
                    .
                    .
                    .
CONECT  724  723 1137                                                        1CLL1440
CONECT  736  735 1137                                                        1CLL1441
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Parsing Files

```
#!/usr/bin/perl

$pdbfile = shift;
($pref  = $pdbfile) =~ s/\.pdb//;

print "Converting $pdbfile to $pref.xyz \n";
open(FILIN, "<$pdbfile" || die "Cannot open pdb file $pdbfile \n ");
open(FILOUT,">$pref.xyz");

while (<FILIN>) {
  if (/^ATOM/) {
    chomp;
    split;
    printf FILOUT "%5d  %4s  %8.3f%8.3f%8.3f\n", $_[1], substr($_[2], 0, 1), $_[5], $_[6], $_[7];
  }
}

close(FILIN);
close(FILOUT);
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Reformating Files

```
./pdb2xyz.pl foo.pdb
```

```
more foo.xyz
```

```
1     N    -6.873 21.082   25.312
2     C    -6.696  22.003  26.447
3     C    -6.318  23.391  25.929
4     O    -5.313  23.981  26.352
5     N    -7.147  23.871  25.013
6     C    -6.891  25.193  24.428
      .
      .
      .
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Executing unix commands inside perl

- Back quotes

```
print `date`;
```

Thu Jun 27 19:06:07 CDT 2002

```
$today = `date`;
print $today;
```

Thu Jun 27 19:06:07 CDT 2002

- System call

```
system("mv $old $new");  # variable substitution done by PERL
system("my_program -abc option_a option_b");
system("ls *.pl  | wc");  # metacharacter expansion done by shell
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Executing applications

```
#!/usr/bin/perl

$pdbfile = shift(@ARGV);
($pref = $pdbfile) =~ s/.pdb//;

system ("rm -r $pref");
system ("mkdir $pref");
chdir ("$pref");

open(SCRIPT,">script");

print SCRIPT "zap\n";
print SCRIPT "load pdb ../$pdbfile\n";
print SCRIPT "background black\n";
print SCRIPT "wireframe off\n";
print SCRIPT "ribbons on\n";
print SCRIPT "color ribbons yellow\n";
for ($i = 0; $i <= 50; ++$i) {
  $name = sprintf("%3.3d",$i);
  print SCRIPT "rotate x 10\n";
  print SCRIPT "write $name.gif\n";
}
print SCRIPT "quit\n";
close SCRIPT;

system("/usr/local/bin/rasmol < script");
system("dmconvert -f mpeg1v -p video ###.gif out.mpeg");
chdir ("..");
```

```
#create a variable $pref using the prefix
#of the pdb filen

#create a directory named after $pref
#change directory into $pref

#create a a file called script

#assigns a value from 0 to 50
#create a file name based on this value
#for every value, rotate 10 degrees
#generate a gif file for each value

#execute the rasmol program
#execute dmconvert to make movie
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

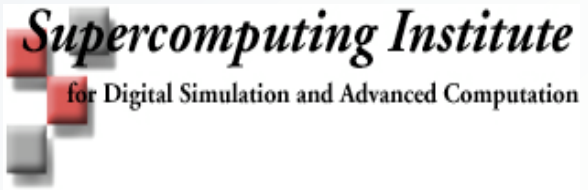
Executing applications

more foo/script

```
background black
wireframe off
ribbons on
color ribbons yellow
rotate x 10
write 000.gif
rotate x 10
write 001.gif
rotate x 10
write 002.gif
.
.
```

ls -lt foo

```
total 99699
-rw-------    1 shamy    support   256504 Oct 21 18:34 out.mpeg
-rw-------    1 shamy    support   995343 Oct 21 18:33 050.gif
-rw-------    1 shamy    support   995343 Oct 21 18:33 049.gif
-rw-------    1 shamy    support   995343 Oct 21 18:33 048.gif
                                      .
                                      .
-rw-------    1 shamy    support   995343 Oct 21 18:32 002.gif
-rw-------    1 shamy    support   995343 Oct 21 18:32 001.gif
-rw-------    1 shamy    support   995343 Oct 21 18:32 000.gif
-rw-------    1 shamy    support     1418 Oct 21 18:32 script
```



submitting jobs to queue

```
#!/usr/bin/perl
# script ll.pl
# usage: ll.pl  arg1  arg2  arg3  arg4

$prefix  = shift;
$program = shift;
$queue   = shift;
$ncpu    = shift;

$script  = "$pref.submit";
$dir = `pwd`;                                  # figure out your current working directory

open(SCRIPT,">$script");
print SCRIPT "# @ initialdir = $dir \n";
print SCRIPT "# @ class = $queue \n";
print SCRIPT "# @ executable = /usr/bin/poe \n";
print SCRIPT "# @ job_type = parallel \n";
print SCRIPT "# @ network.MPI = css0,shared,US \n";
print SCRIPT "# @ tasks_per_node = 1 \n";
print SCRIPT "# @ node = $ncpu \n";
print SCRIPT "# @ arguments = $program < $prefix.inp \n";
print SCRIPT "# @ output    = $prefix.out \n";
print SCRIPT "# @ error     = $prefix.err \n";
print SCRIPT "# @ notification = never \n";
close SCRIPT;

system("llsubmit $script");
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

submitting jobs to queue

%ll.pl job program sp_queue 2

%more job.script

```
# @ initialdir = /home/msia/shamy/perl
# @ class = sp_queue
# @ executable = /usr/bin/poe
# @ job_type = parallel
# @ network.MPI = css0,shared,US
# @ tasks_per_node = 1
# @ node = 2
# @ arguments = program < job.inp
# @ output     = job.out
# @ error      = job.err
# @ notification = never
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Submitting jobs to queue (Creating scripts with templates)

```
#!/usr/bin/perl
# script ll.pl
# usage: ll.pl  arg0  arg1  arg2  arg3

$prefix    = shift;
$program = shift;
$queue   = shift;
$ncpu    = shift;

$script  = "$pref.submit";
$dir = `pwd`;

open(TEMPLATE,"<ll.template");
open(SCRIPT,">$script");

While (<TEMPLATE>) {
      s/prefix/$prefix/;
      s/directory/$dir/;
      s/program/$program/;
      s/queue/$queue/;
      s/ncpu/$ncpu/;
      print SCRIPT;
}

system("llsubmit $script");
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Submitting jobs to queue (Creating scripts with templates)

more ll.template

```
# @ initialdir = directory
# @ class = queue
# @ executable = /usr/bin/poe
# @ ob_type = parallel
# @ network.MPI = css0,shared,US
# @ tasks_per_node = 1
# @ node = ncpu
# @ arguments = program < prefix.inp
# @ output    = prefix.out
# @ error     = prefix.err
# @ notification = never
```

Supercomputing Institute for Digital Simulation and Advanced Computation

Exit status & file status

- Exit status of last pipe, system command, or `` (backquotes)

```
@output = `date`;
print "Exit status: $?\n";  # exit status is 0 if no errors
```

- File creation, modification, last access dates, other status info

```
($dev, $ino, $mode, $nlink, $uid,
 $gid, $rdev, $size, $atime,
 $mtime, $ctime, $blksize, $blocks ) = stat($filename);
```

- Example

```
($atime, $mtime) = (stat($filename))[8,9];
unlink($filename) unless $atime < 2592000  # 30 days = 3600 * 2 * 30
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting

- Files
- Lines within files
- Occurrences of strings in files or file names
- Complex pattern matches

Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting

```
#!/usr/bin/perl
my $characters  =  0;
my $words       =  0;
my $lines       =  0;
my $line_length  =  0;
my $paragraphs  =  0;
my $word        =  "";

    while(<>) {
        $line_length  = length($_);
        $characters += $line_length;
        $lines++;
        $paragraphs++ if($line_length == 1);

        for $word (split) {  $words++;  }
    }
    $paragraphs++;
    printf "%8d Chars\n", $characters;
    printf "%8d Words\n", $words;
    printf "%8d Lines\n", $lines;
    printf "%8d Paragraphs\n", $paragraphs;
exit;
```

wc.pl text

```
     531 Chars
      94 Words
       9 Lines
       1 Paragraphs
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting

```
#!/usr/bin/perl
#simple_frequency.pl
my $characters  =  0;
my $words       =  0;
my $lines       =  0;
my $line_length =  0;
my $paragraphs  =  0;
my $uniq_words  =  0;
my $word        = "";
my %wordhash;
while(<>) {
        $line_length  = length($_);
        $characters += $line_length;
        $lines++;
        $paragraphs++ if($line_length == 1);
        for $word (split) {
                $words++;
                $wordhash{lc($word)}++;
        }
}
$paragraphs++;
$uniq_words = keys %wordhash;

printf "%8d Chars\n",       $characters;
printf "%8d Words\n",       $words;
printf "%8d Unique words\n", $uniq_words;
printf "%8d Lines\n",       $lines;
printf "%8d Paragraphs\n",   $paragraphs;
print "\n";

print "Word frequency counts\n";
print "=====================\n";
foreach $i (keys(%wordhash)) {
        printf "%8d %s\n", $wordhash{$i}, $i;
}
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting: output

```
simple_frequency.pl text
     531 Chars
      94 Words
      62 Unique words
       9 Lines
       1 Paragraphs

Word frequency counts
=====================
       1 through
       4 the
       1 help-info
       1 tells
       2 search.
       1 keyword
       2 desktop,
       1 information
       1 (isp).
       1 provider
       1 1:
       1 3:
       1 5:
       3 your
       1 7:
       1 silicon
       1 9:
       2 from
       2 toolchest
       2 search
       1 provider.
           ...
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting example with options

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

my $characters  =  0;
my $lines       =  0;
my $line_length =  0;
my $words       =  0;
my $paragraphs  =  0;
my $uniq_words  =  0;
my $word        =  "";
my %wordhash;

# usage: wfc [-a | -d | -r ] file [file ...]
    %tool_box = (
            "-a"  =>  \&alphabetic_list,
            "-d"  =>  \&descending_frequency,
            "-r"  =>  \&reverse_dictionary,
            "-"   =>  \&none
    );

    $action = ( $ARGV[0] =~ /^-/ )  ?  shift  :  "-a";
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting example with options

```
while(<>) {

        $line_length = length($_);
        $characters += $line_length;
        $lines++;
        $paragraphs++ if($line_length == 1);

        $_ =~ s/[\.,\?\[\]\!\@\#\$\%\^\&\*\(\)\+=\":;<>]//g;

        foreach $word (split) {
                $words++;
                $wordhash{lc($word)}++;
        }
}
$paragraphs++ if $lines;

$uniq_words = keys %wordhash;
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting example with options

```
    printf "%8d Chars\n",       $characters;
    printf "%8d Words\n",       $words;
    printf "%8d Unique words\n", $uniq_words;
    printf "%8d Lines\n",       $lines;
    printf "%8d Paragraphs\n",  $paragraphs;
    print  "\n";

    if( defined $tool_box{$action} ) {
            $tool_box{$action}->();
    }
exit;

sub none {}
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting example with options

```
sub alphabetic_list {
        print "Alphabetic list of word frequency counts\n";
        print "=======================================\n";
        foreach $i ( sort keys %wordhash ) {
                printf "%8d %s\n", $wordhash{$i}, $i;
        }
}

sub decending_frequencey {
        print "Word frequency counts, decending order\n";
        print "=====================================\n";
        foreach $i ( sort { $wordhash{$b} <=> $wordhash{$a} } keys %wordhash ) {
                printf "%8d %s\n", $wordhash{$i}, $i;
        }
}

sub reverse_dictionary {
        print "Reverse dictionary word frequency counts\n";
        print "========================================\n";
        foreach $i ( sort { reverse($a) cmp reverse($b) } keys %wordhash ) {
                printf "%8d %s\n", $wordhash{$i}, $i;
        }
}
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Counting example with options

```
command:
wfc -d text
     531 Chars
      91 Words
      59 Unique words
       9 Lines
       1 Paragraphs

Word frequency counts, descending order
=======================================
       5 on
       4 the
       4 search
       3 your
       3 you
       3 to
       3 a
       2 service
       2 choose
       2 desktop
       2 toolchest
       2 an
       2 internet
       2 connect
       2 how
       2 answers
       2 quick
    ...
```

Supercomputing Institute
for Digital Simulation and Advanced Computation

Mail

- Sending mail
  - use Mail::Mailer when you can
  - otherwise use sendmail on UNIX systems
  - Location varies: /usr/local/, /usr/lib/, /usr/sbin/, ...
- Processing contents from a file
- Processing received mail

Supercomputing Institute
for Digital Simulation and Advanced Computation

Mail example

```
#!/usr/bin/perl
    my $output = `date`;
    print "Output: $output";

    open( MAIL, "|/usr/lib/sendmail -oi -t") or die "Can't fork for sendmail: $!\n";

    print MAIL "From: \"Yuk Sham\" <shamy\@msi.umn.edu>\n";
    print MAIL "To: \"Yuk Sham\" <shamy\@msi.umn.edu>\n";
    print MAIL "Subject: Sending mail with PERL\n";
    print MAIL "\n";   ######## DON'T FORGET THIS ONE!!!!!

print MAIL <<"EOF";
The body of the message goes here.
...
And here...
EOF
close(MAIL) or warn "sendmail did not close properly";
exit;
```


Supercomputing Institute
for Digital Simulation and Advanced Computation

Report generation

- Sort files
- Extract selected data & store in arrays or hashes
- Sort
- Output
  - Format, paginate, print
  - Generate HTML pages
  - Store/update DBM files (Berkeley data base package)

Supercomputing Institute
for Digital Simulation and Advanced Computation

More info

- CPAN - Comprehensive Perl Archive Network
  - http://www.cpan.org
  - Source, binaries, libs, scripts, FAQ's, links
- Perl Resource Topics
  - http://www.perl.com/pub/q/resources
- Others
  - http://www.netcat.co.uk/rob/perl/win32perltut.html
  - http://www.1001tutorials.com/perltut/index.html
  - http://www.perlmasters.com/tutorial
  - http://www-2.cs.cmu.edu/cgi-bin/perl-man
  - Countless more are available...

Supercomputing Institute
for Digital Simulation and Advanced Computation

Contact the Institute for additional help

Cal Kirchhof
Visualization Consultant
Phone: (612) 625 0056 (direct)
Email: cal@msi.umn.edu

Yuk Sham
Computational Biology/Biochemistry Consultant
Phone: (612) 624 7427 (direct)
Email: shamy@msi.umn.edu

Supercomputing Institute
for Digital Simulation and Advanced Computation