Slide 1

CSE2395/CSE3395

Topic 9

Processes and formats

Learning Perl 2nd edition
chapter 14, pages 142-152
chapter 11, pages 116-128
Programming Perl 3rd edition
pages 234-241, 747-755, 426-428
Programming Perl 2nd edition
pages 191-195, 168, 121-127

`perlform`, `perlopentut` manpages

1

Slide 2

Last time

Covered in Topic 8

CSE2395/CSE3395

- Files
  ‣ opening and closing
  ‣ reading from
    – <HANDLE>
  ‣ writing to
    – print HANDLE
  ‣ testing
    – -f, -d, -x, etc.
  ‣ renaming, deleting
- Directories
  ‣ reading directory entries
    – opendir, readdir
  ‣ creating, removing

2

Slide 3

To be covered today

CSE2395/CSE3395

- Processes
  ‣ invoking other programs with `system`
  ‣ talking to other programs
    – backquotes
    – opening a pipe with `open`
- Formats
  ‣ defining with `format`
  ‣ using with `write`

3

Slide 4

Running a program from Perl

system

CSE2395/CSE3395

- `system` function invokes another program (process)
  ‣ `system "date";`
  ‣ `system "ls -l $ENV{'HOME'}";`
- Perl program waits for invoked process to finish before continuing
- Returns exit status of invoked process
- Invoked process inherits from Perl program
  ‣ filehandles (STDIN, STDOUT, STDERR)
  ‣ environment (%ENV)

Llama2 pages 142-145; *Camel3* pages 811-812
Camel2 page 230; `perlfunc` manpage

4

Slide 5

Capturing output from programs

The backquote (` `` `) operator

CSE2395/CSE3395

- Like `system`, except return value is the invoked process' standard output
  ‣ `$now = `date`;`
  ‣ trailing newline character of `date`'s output is kept
- Variables are interpolated inside backquotes as with double-quoted strings
- In list context, returns each line as a separate element
  ‣ `@files_in_home_dir = `ls -1 $ENV{'HOME'}`;`

Llama2 pages 145-146; *Camel3* page 80
Camel2 pages 52-53; `perlop` manpage

5

Slide 6

Getting more control

Overcoming limitations of system and backquotes

CSE2395/CSE3395

- With both `system` and backquotes, Perl program must wait until invoked process finishes
  ‣ possibly a long time
- With backquotes, output of invoked process is stored all at once
  ‣ possibly a lot of data
- Sometimes better to deal with each line of process' output as it comes
- Solution is to invoke the process as a pipe with `open` function

6

Using open to start processes

Reading from pipes

- To open process to read from, place vertical bar (|) at end of string
  - ‣ `open FILELIST, "ls -1|";`
  - ‣ "ls -1" is not a file to read, but a program to run
- Perl invokes command `ls -1` and attaches its standard output to `FILELIST` filehandle
- Reading from `FILELIST` reads one line from output of invoked process
  - ‣ `$filename = <FILELIST>; # Get a filename.`

Llama2 pages 146-147; *Camel3* pages 750-752
Camel2 pages 192-193
`perlfunc, perlopentut` manpages

7

Using open to start processes

Writing to pipes

- To open process to write to, place vertical bar (|) at start of string
  - ‣ `open PRINTER, "|lpr -Plj4";`
  - ‣ "lpr -Plj4" is not a file, but a program to run
- Perl invokes command `lpr -Plj4` and attaches its standard input to `PRINTER` filehandle
- Writing to `PRINTER` writes to input of invoked command
  - ‣ `print PRINTER "Hello, printer lj4!\n";`

Llama2 pages 146-147; *Camel3* pages 750-752
Camel2 pages 192-193
`perlfunc, perlopentut` manpages

8

Example

Sending a quotation by email

```perl
# Assume that "fortune" and "mail" exist.
# Probably should check return values of these:
open QUOTE, "/usr/games/fortune|";
open MAIL, "|/bin/mail -s Quote foo@bar.com";

print MAIL "Here's a little quotation:\n";
# Read lines from the "fortune" process.
while (<QUOTE>)
{
    # Print quotation, indented with "> ", to
    # the "mail" process.
    print MAIL "> $_";
}
# Explicitly closing a pipe makes Perl wait
# for the process to finish.
# We ought to check return values of these too:
close MAIL;
close QUOTE;
```

9

Formats

Printing things out nicely

- Perl can generate simple formatted reports from data
- Known as formats

`data` →

```
May                HAP 7821
J. Random Hacker
1131 Nonesuch Drive
Nowhere            FRZ 0123
Ellen M. O'Pea
Cloud 9
Utopia Heights SQU 9876
L. B. Quirky
61 Vapour Street
Steamville      JPK 0234
Sharon Rock
```

Llama2 page 116; *Camel3* pages 234-241
Camel2 pages 121-123; `perlform` manpage

10

Formats

Perl's implementation

- A format is defined using the `format` keyword
  - ‣ defines the fixed and variable parts of the output
  - ‣ associates variables or expressions with fields in the format
- A format is invoked using the `write` function
  - ‣ prints one record using the current values of the variables named in the filehandle's associated format

Llama2 pages 116-118; *Camel3* pages 234-237
Camel2 pages 121-123; `perlform` manpage

11

Defining a format

Using the format keyword

Required syntax

Name of format; usually matches name of filehandle

Value lines
- name the global variables which will be substituted into fields on above field lines

```
format ADDRESSLABEL =
----------------------------
@<<<<<<<<<<<<<<<<<<<<<<<<<<<
$name
@<<<<<<<<<<<<<<<<<<<<<<<<<<<
$street
@<<<<<<<<<<<<<< @<< @>>>
$city, $state, $code
.
```

Field lines
- constant lines have no matching value lines
- @ indicates variable field value
- > or < indicates justification

Dot on line of its own ends format

Llama2 pages 117-118; *Camel3* pages 234-236, 716-717;
Camel2 pages 122-124, 168; `perlform` manpage

12

Field types

A selection

- @<<< or @||| or @>>>
  - left-, centre- or right-justified with spaces, to a width of four characters
  - values longer than this are truncated
  - For wider fields, add more < or | or >
- @###.##
  - numeric values, aligned on decimal point
  - four places to left of decimal point, two to right
- @*
  - multi-line value

  Llama2 pages 120-124; *Camel3* pages 235-237
  Camel2 pages 122-123; `perlform` manpage

13

Invoking a format

The `write` function

- `write` function takes name of filehandle (*not* a format) to print a record to
  - `write ADDRESSLABEL;`
  - filehandle's default format is the format with the same name
    - can be changed through the special `$~` variable and `select` function
  - to write a record to standard output, use `write STDOUT` (uses the `STDOUT` format by default)
  - the record is printed using the current values of the variables named in the format definition

  Llama2 pages 118-120, 125-126; *Camel3* pages 829, 664, 780-781; *Camel2* pages 242, 129-131, 210-211 `perlform` manpage

14

Example

Printing address labels

```
format ADDRESSLABEL =
---------------------------
@<<<<<<<<<<<<<<<<<<<<<<<<<<<
$name
@<<<<<<<<<<<<<<<<<<<<<<<<<<<
$street
@<<<<<<<<<<<<<<< @<< @>>>
$city,        $state, $code
.

# Assume file "addresses" has lines like this:
# name:street:city:state:postcode
open (ADDRESSES, "<addresses");
open (ADDRESSLABEL, "|lpr -Plj4");
while(<ADDRESSES>) {
   chomp;
   # Set the variables used in the format.
   ($name, $street, $city, $state, $code) = split /:/;
   write ADDRESSLABEL;
}
```

15

Printing headers

The top-of-page format

- Can define a header which is printed at top of every page of output
  - page length defaults to 60 lines, but can be changed with special variable `$=` and `select` function
  - prints special top-of-page format before printing next record, if record would not fit on current page
  - top-of-page format's default name is made by appending "`_TOP`" to filehandle name
    - e.g., `ADDRESSLABEL_TOP`
    - can change format used with special `$^` variable and `select` function
  - special variable `$%` contains current page number

  Llama2 pages 124, 125-127; *Camel3* pages 829, 664, 780-781; *Camel2* pages 242, 129-131, 210-211; `perlform` manpage

16

Example

Pretty-printing `/etc/passwd`

```
format STDOUT_TOP =
Username     UID Real name       (page @>)
                                      $%
-----------------------------------------
.

format STDOUT =
@<<<<<<<<< @>>>> @<<<<<<<<<<<<<<<<<<<<<<<<
$user,     $uid, $real
.

open PASSWD, "</etc/passwd";

while (<PASSWD>)
{
   ($user, $uid, $gcos) = (split /:/)[0,2,4];
   ($real) = split /,/, $gcos;
   write STDOUT;
}
```

17

Covered today

- Processes
  - invoking other programs with `system`
  - talking to other programs
    - backquotes
    - opening a pipe with `open`
- Formats
  - defining with `format`
  - using with `write`

18

Going further

- `IPC::Open2`
  - ‣ what you need when you try to `open HANDLE,`
    `"|command|"`
  - ‣ *Camel3* pages 900-901; *Camel2* pages 455-457
- `IO::Socket`
  - ‣ communicating with processes across the Internet
  - ‣ *Camel3* pages 899-900
- Threads
  - ‣ A lightweight alternative to processes
  - ‣ *Camel3* pages 446-463

CSE2395/CSE3395

19

Next time

- Common Gateway Interface (CGI)
  - ‣ making interactive pages on the World Wide Web

CSE2395/CSE3395

Reading:
Learning Perl 2nd edition chapter 19, pages 180-209
Programming Perl 3rd edition page 878
`CGI` manpage

20

CSE2395/CSE3395

21