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

Last time

Covered in Topic 8

- 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

To be covered today

- Processes
- ▶ invoking other programs with system
- ▶ talking to other programs
- back quotes
- opening a pipe with open
- Formats
- ▶ defining with format
- ▶ using with write

Running a program from Perl

- 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 (%FNV)

Llama2 pages 142-145; Camel3 pages 811-812 Camel2 page 230; perl func manpage

4

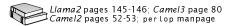
Capturing output from programs

The backquote (``) operator

- Like system, except return value is the invoked process' standard output
 - \$ now = `date`;

CSE2395/CSE3395

- 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
 - afiles in home dir = `ls -1 \$ENV{'HOME'}`;



Getting more control

Overcoming limitations of system and backguotes

- 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 (1) at end of string
 - ▶ open FILELIST, "ls -1|";

CSE2395/CSE3395

CSE2395/CSE3395

CSE2395/CSE3395

close QUOTE;

- → "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

Using open to start processes

- To open process to write to, place vertical bar (I) 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 Sperlfunc, perlopentut manpages

8

Example

Sending a quotation by email

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:

Formats Printing things out nicely Perl can generate simple formatted reports from data J. Random Hacker 1131 Nonesuch Drive Nowhere FRZ 0123 ■ Known as formats CSE2395/CSE3395 Ellen M. O'Pea Cloud 9 Utopia Heights SQU 9876 data L. B. Quirky 61 Vapour Street Steamville JPK 0234 Llama2 page 116; Camel3 pages 234-241 Camel2 pages 121-123; perl form manpage

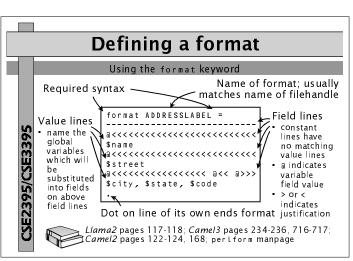
10

Formats

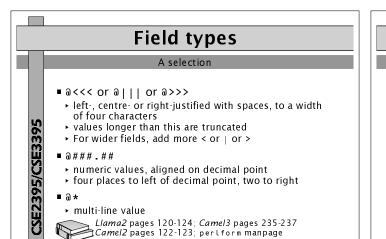
Perl's implementation

- A format is defined using the format
 - defines the fixed and variable parts of the output
 - ▶ associates variables or expressions with fields in the
- 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





11 12



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



13

14

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; pertform manpage

16

Covered today

- Processes
 - $\,\blacktriangleright\,$ invoking other programs with $\,$ system
- ► talking to other programs
 - backquotes
- opening a pipe with open
- Formats

 defining

 using w
 - ightharpoonup defining with format
 - ▶ using with write

17 18

Going further

More things related to today's topic

- IPC::0pen2
- ightharpoonup 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 InternetCamel3 pages 899-900

- ► A lightweight alternative to processes
- Camel3 pages 446-463

Next time To be covered in Topic 10 ■ 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

19 20

CSE2395/CSE3395 lecture notes copyright © 2000-2001 Deborah Pickett. Reproduction of this presentation for nonprofit study use is permitted. All other reproduction, including for other educational courses, must be authorized in writing by the author.

CSE2395/CSE3395