Perl

A language for Systems and Network Administration and Management

What is Perl?

- Perl is a programming language
- The best language for processing text
- Cross platform, free, open
- Microsoft have invested heavily in ActiveState to improve support for Windows in Perl
- Has excellent connection to the operating system
- Has enormous range of modules for thousands of application types
-

What is Perl? 2

- Robust and reliable (has very few bugs)
- Supports object oriented programming
- Good for big projects as well as small
- Java 1.4 has borrowed one of Perl's best features: *regular expressions*
- Perl has garbage collection
- The "duct tape of the Internet"
- Easy to use, since it usually "does the right thing"
- Based on freedom of choice: "There is more than one way to do it!"
- TIMTOWTDI

•

•

Compiled and run each time

- Perl is interpreted, but runs about as fast as a Java program
- Software development is very fast
- The Apache web server provides mod_perl, allows Perl applications to run very fast
- Used on some very large Internet sites:
- The Internet Move Database
- Macromedia, Adobe, http://slashdot.org/

•

Perl is Evolving

- Perl 6 will introduce many great features to make Perl
- easier to use
- Even more widely usable for more purposes
- Even better for bigger projects

•

Eclectic

- Borrows ideas from many languages, including:
- C, C++
- Shell
- Lisp
- BASIC

- Fortran
- Many others...
-

Regular Expressions

- One of the best features of Perl
- A new concept for most of you
- very useful!
- Used to:
- extract information from text
- transform information
- You will spend much time in this topic learning about regular expressions
-

Why should I learn it?

- It will be in the final exam!
- Okay, that's to get your attention, but...
- Consider a real-life sys-admin problem:
- You must make student accounts for 1500 students
- TEACHING BEGINS TOMORROW!!!
- The Computing Division has a multi-million dollar application to give you student enrollment data
- it can only give you PDF files with a strange and irregular format for now (But Oh, it will be infinitely better in the future! Just wait a year or two...)
-

The available data

- Has a variable number of lines before the student data begins
- Has a variable number of columns between different files
- Has many rows per enrolled student
- Goes on for dozens of pages, only 7 students per page!!!!!!!
- There are two formats, both equally peculiar!!!!
-

Sample data for new courses:

- 15 N CHAN Wai Yee F 993175560 H123456(5) 28210216 CHEUNG
- 10-SEP-01 10-SEP-01 21234567 WAI CHI SISTER 91234567
-
-

Problems

- There is a different number of lines above the student records
- There is a different number of characters within each column from file to file
- There are many files
- The format can change any time the computing division determines necessary
-

Solution in Perl

- #! /usr/bin/perl -w
-
- use strict;
-
- my $course;
- my $year;
-

- while (<>)
- {
- chomp;
-
- if (/^\s*Course :\s(\d+)\s/)
- {
- $course = $1;
- undef $year;
- next;
- }
- elsif (m!^\s*Course :\s(\d+)/(\d)\s!)
- {
- $course = $1;
- $year = $2;
- next;
- }
- if (
- my ($name, $gender, $student_id, $hk_id)
- #= m!\s\s+([A-Z]+(?: [A-Z][a-z]*)+)\s\s+([MF])\s+(\d{9})\s\s+([a-zA-Z]\d{6}\([\dA-Z]\))!)
- = m{
- \s\s+ # at leaset 2 spaces
- (# this matches $name
- [A-Z]+ # family name is upper case
- (?:\s[A-Z][a-z]*)+ # one or more given names
-)
- \s\s+ # at leaset 2 spaces
- ([MF]) # gender
- \s+ # at least one space
- (\d{9}) # student id is 9 digits
- \s\s+ # at leaset 2 spaces
- ([a-zA-Z]\d{6}\([\dA-Z]\)) # HK ID
- }x
-)

- {
- print "sex=$gender, student ID = $student_id, ",
- "hkID = $hk_id, course = $course, name=$name, ",
- defined $year ? "year = $year\n" : "\n";
- next;
- }
- warn "POSSIBLE UNMATCHED STUDENT: $_\n" if m!^\s*\d+\s+!;
- }
-

But I can use any other language!

- I will give you HK$200 if you are the first person to write a solution in another language in fewer keystrokes
- Note: the Perl solution given has:
- comments
- Plenty of space to show structure
- handles exceptional situations (i.e., it is robust)
- To claim your $200 from Nick, your solution must have
- similar space for comments
- Similar readability and robustness
- Be written in a general purpose language using ordinary libraries
-

Any other solution may take longer to write

- This program took a very short time to write
- It is very robust
- For problems like this, Perl is second to no other programming language.
-

The hello world program

- print “hello world\n”
-

Variables

- There are three basic types of variable:
- ***Scalar*** (can be a number or string or. . .)
- ***Array*** (an ordered array of scalars)
- ***Hash*** (an unordered array of scalars indexed by strings instead of numbers)
- Each type distinguished with a “funny character”
-

$Scalars:

- Start with a dollar sign
- Hold a single value, not a collection
- A string is a scalar, so is a number
- Examples:
- $apple = 2;
- $banana = “curly yellow fruit”;
-
-

@Array

- Starts with a @
- Indexes start at 0, like in C
-

•

•

%Hashes

• Unfamiliar concept to many of you

• Like an array, but indexed by a string

• A data structure like a database

•

Conclusion

• Perl is optimised for text and systems administration programming

• Has great portability

• Is strongly supported by Microsoft

• Has three main built-in data types:

• Scalar: starts with $

• Array: starts with @

• Hash: starts with %

•