Perl

Ming-Hwa Wang, Ph.D.
COEN 388 Principles of Computer-Aided Engineering Design
Department of Computer Engineering
Santa Clara University

Scripting Languages

- interpreter vs. compiler
  - efficiency concerns
- strong typed vs. type-less
- Perl: Practical Extraction and Report Language
  - pattern matching capability

Resources

- Unix Perl manpages
- Usenet newsgroups: comp.lang.perl
- Perl homepage: http://www.perl/com/perl/, http://www.perl.org
- Download: http://www.perl.com/CPAN/

To Run Perl Script

- run as command: perl –e 'print "Hello, world!\n";'
- run as scripts (with chmod +x): perl <script>

```
#!/usr/local/bin/perl –w
use strict;
print "Hello, world!\n";
exit 0;
```

- switches

--	terminates switch processing
-0<oct_name>	specifies the record separator ($/) as an octal number
-a	truns on autosplit mode when used with a –n or -p
-c	causes perl to check the syntax
-d	runs the script under the Perl debugger
-d:<lib_name>	runs the script under the control of a debugging/tracing module installed in Perl library as Devel::<lib_name>, e.g., Devel::DProf for profiler
-D<number | list>	set debugging flags
-e commandline	to enter one or more lines of script
-F<pattern>	Specifies the pattern to split on if –a is used
-h	help
-i<extension>	Specifies that file processed by the < > construct are to be edited in-place
-I<directory>	directories specified by –I are prepended to @INC
-l<oct_name>	enable automatic line-end processing
-(m|M)[-]<module>[=arg{,arg}]	executes use <module> before executing your script
-n -p	causes Perl to assume a loop around your script
-P	causes your script through the C preprocessor before compilation
-s	enable switch parsing
-S	makes Perl use the PATH environment variable to search for the script
-T	forces taint checks to be turned on so you can test them
-u	causes Perl to dump core after compiling your script
-U	allow Perl to do unsafe operations
-v	print version
-V	print Perl configuration and @INC values
-V:<name>	print the value of the named configuration value
-w	print warning for unused variable, using variable without setting the value, undefined functions, references to undefined filehandle, write to filehandle which is read-only, using a non-number as it were a number, or subroutine recurse too deep, etc.
-x<directory>	tells Perl to extract a script that is embedded in a message

- debug: perl –d <script>
  - debugger commands:

commands	usage
h [command]	for help message
p expr	print the expression
x expr	execute and print the expression
V [pkg[vars]]	display variables
X [vars]	same as V currentpkg[vars]
T	produce stack backtrace
n	next
<CR>	repeat last n or s command
c [line]	continue [break on line only once]
- or l [min+incr|min-max|line|subname]	list next few lines
w [line]	list window a few lines [around the line]

.	return debugger pointer to the last executed line and print it out
f filename	switch to view a different file
/pattern/, ?pattern?	search forward/backword for pattern
L	list all break points or actions for current file
S[![pattern]	list subroutine names [not]matching pattern
t	toggle trace mode
t expr	trace through execution of expr
b [subname] [line] [condition]	set a breakpoint [in subname] [at line] [when condition]
d [line]	delete a breakpoint [at line]
D	delete all installed breakpoints
a [line] command	set an action to be done before the line is executed
A	delete all installed actions
O [opt[=val]]	set or query values of options
< command	set an action to happen before every debugger prompt
> command	set an action to happen after every debugger prompt
! [-] number	redo a previous commands
! pattern	redo last command that started with pattern
!! command	run command in a subprocess
H -number	display last number command
q or ^D	quit
R	restart
| dbcmd or || dbcmd	run debugger cmd piping DB::OUT to $ENV{PAGER}
= [alias value]	define a command alias or list current aliases

Pragmas

- syntax: use *module*;
- miscellaneous

Benchmark	check and compare running times of code
Config	access Perl configuration information
Env	import environment variables
English	use English or awk names for punctuation variables
Getopt::Long	extended processing of command-line options
Getopt::Std	process single-character switches with switch clustering
lib	manipulate @INC at compile time
Shell	run shell commands transparently within Perl
strict	restrict unsafe constructs
Symbol	generate anonymous globs; qualify variable names
subs	predeclare subroutine names
vars	predeclare global variable names

- error handling and logging

Carp	generate error messages
diagnostics	force verbose warning diagnostics
sigtrap	enable stack backtrace on unexpected signals
Sys::Syslog	Perl interface to UNIX syslog(3) calls

- file access and handling

Cwd	get pathname of current working directory
DirHandle	supply object methods for directory handles
File::Basename	parse file specifications
File::CheckTree	run many tests on a collection of files
File::Copy	copy files or filehandles
File::Find	traverse a file tree
File::Path	create or remove a series of directories
FileCache	keep more files open than the system permits
FileHandle	supply object methods for filehandles
SelectSaver	save and restore selected filehandles

- test processing and screen interfaces

Pod::Text	convert POD data to formatted ASCII text
Search::Dict	search for key in dictionary file
Term::Cap	terminal capabilities interface
Term::Complate	word completion module
Text::Abbrev	create an abbreviation table from a list
Text::ParseWords	parse text into a list of tokens
Text::Soundex	the soundex algorithm described by Knuth
Text::Tabs	expand and unexpand tabs
Text::Wrap	wrap text into a paragraph

- database interfaces

AnyDBM_File	provide framework for multiple DBMs
DB_File	tied access to Berkeley DB
GDBM_File	tied access to GDBM library
NDBM_File	tied access to NDBM files
ODBM_File	tied access to ODBM files
SDBM_File	tied access to SDBM files

- mathematics

integer	do arithmetic in integer instead of double
Math::BigFloat	arbitrary-length floating-point math package
Math::BigInt	arbitrary-length integer match package
Math::Complex	complex numbers package

- networking and interprocess communication

IPC::Open2	open a process for both reading and writing
IPC::Open3	open a process for reading, writing, and error handling

Net::Ping	check whether a host is online
Socket	load the C socket.h defines and structure manipulators
Sys::Hostname	try every conceivable way to get hostname

- time and locale

Time::Local	efficiently computer time from local and GMT time
I18N::Collate	compare 8-bit scalar data according to the current locale

- autoloading and dynamic loading (for developers)

AutoLoader	load functions only on demand
AutoSplit	split a module for autoloading
Devel::SelfStubber	generate stubs for a SelfLoading module
DynaLoader	automatic kynamic loading of Perl modules
SelfLoader	load functions only on demand

- language extensions and platform development support (for developers)

ExtUtils::Install	install files from here to there
ExtUtils::Liblist	determine libraries to use and how to use them
ExtUtils::MakeMaker	create a Makefile for a Perl extension
ExtUtils::Manifest	utilities to write and check a MANIFEST file
ExtUtils::Miniperl	write the C code for perlmain.c
ExtUtils::Mkbootstrap	make a bootstrap file for use by Dynaloader
ExtUtils::Mksymlists	write linker option files for dynamic extension
ExtUtils::MM_OS2	methods to override UNIX behavior in ExtUtils::MakeMaker
ExtUtils::MM_Unix	methods used by ExtUtils::MakeMaker
ExtUtils::MM_VMS	methods to override UNIX behavior in ExtUtils::MakeMaker
fcntl	load the C fcntl.h defines
POSIX	interface to IEEE Std 1003.1
Safe	create safe namespaces for evaluating Perl code
Test::Harness	run Perl standard test scripts with statistics

- object-oriented programming support (for developers)

Exporter	default import method for modules
overload	overload Perl's mathematical operations
Tie::Hash	base class definitions for tied hashes
Tie::Scalar	base class definitions for tied scalars
Tie::StdHash	base class definitions for tied hashes
Tie::StdScalar	base class definitions for tied scalars
Tie::SubstrHash	fixed-table-size, fixed-key-length hashing

Variables

- $: scalar variable
- @: array variable
- %: associated array (or hash) variable
- &: subroutine
- *: typeglob (references or alias for passing or storing filefolders or symbol tables)

Predefined variables

- global special variables
  $_($ARG), $.($INPUT_LINE_NUMBER or $NR), $/($INPUT_RECORD_SEPARATOR or $RS), $,($OUTPUT_FIELD_SEPARATOR or $OFS), $\($OUTPUT_RECORD_SEPARATOR or $ORS), $"($LIST_SEPARATOR), $;($SUBSCRIPT_SEPARATOR or $SUBSEP), $:($FORMAT_LINE_BREAK_CHARACTER), $#($OFMT), $[(the index of the first element in an array, and of the first character in a substring)
- regular expression special variables
  $&($MATCH), $'($POSTMATCH), $`($PREMATCH), $+($LAST_PATTERN_MATCH), $*($MULTILINE_MATCHING)
- Perl filehandle special variables
  $|($OUTPUT_AUTOFLUSH), $=($OUTPUT_LINE_PER_PAGE), $%($FORMAT_PAGE_NUMBER), $-($FORMAT_LINE_LEFT), $~($FORMAT_NAME), $^($FORMAT_TOP_NAME)
- system special variables
  $$($PROCESS_ID or $PID), $]($PERL_VERSION), $0($PROGRAM_NAME), $@($EVAL_ERROR), $!($OS_ERROR), $?($CHILD_ERROR), $<($REAL_USER_ID or $UID), $(($REAL_GROUP_ID or $GID), $)($EFFECTIVE_GROUP_ID or $EGID), $>($EFFECTIVE_USER_ID or $EUID)
- other global special variables
  $^A($ACCUMULATOR), $^D($DEBUGGING), $^F($SYSTEM_FD_MAX), $^H(internal compiler hints enabled by pragmatic modules), $^I($INPLACE_EDIT), $^L($FORMAT_LINEFEED), $^O($OSNAME), $^P($PERLDB), $^T($BASETIME), $^W($WARNING), $^X($EXECUTABLE_NAME)
- global special arrays
  $ARGC, @ARGV, $#ARGV, @INC(for include paths), @F(when –a option is used), %INC(included headers), %ENV, %SIG
- global special filehandles
  STDIN, STDOUT, STDERR, __LINE__, __FILE__, __END__, __DATA__

Values

- assign values: assign scalar to scalar variable, assign array/hash to array/hash variable, assign array/hash to scalar variable
- assign multiple variables, cascade assignment
- Numeric: integer, float, scientific notation, hexadecimal, octal, underline for legibility
- String literal: double-quote for interpolation, single-quote, \n, \r, \t, \f, \b (backspace), \a (bell), \e (ESC), \cC (control-C), \u (force next char

to upper), \l, \U (for all following chars to upper), \L, \Q (backslash all following non-alphanumeric chars), \E (end for \U, \L, or \Q)

customary	generic	meaning	interpolation
‘ ‘	q//	literal	no
“ “	qq//	literal	yes
` `	qx//	command	yes
( )	qw//	word list	no
//	m//	pattern match	yes
s///	s///	substitution	yes
y///	tr///	translation	no

File handles

- read: open(FID, “file_name”); open(FID, “<file_name”);
- write: open(FID, “>file_name”);
- append: open(FID, “>>file_name”);
- output filter: open(FID, “| output_pipe_command”);
- input filter: open(FID, “input_pipe_command |”);
- read from file: <STDIN>
- write to file: print FID …;
- close file: close(FID);

Operators

- arithmetic operator: +, -, *, /, %, **
- string operator: ., x
- logic operator: &&, and, ||, or, !, not
- comparison operator:
  - numeric: ==, !=, <, >, <=, >=, <=>
  - string: eq, ne, lt, gt, le, ge, cmp
- assignment operator: =, op=
- autoincrement/autodecrement (either prefix or postfix): ++, --
- file test operator: -e for exist, -r for readable, -w for writable, -d for directory, -f for file, -T for text file
- list operator: sort, reverse, reverse sort
- pointer operator: reference ->, dereference \
- array operator: [ ], [..]
- hash operator: { }, { , }
- number operator: $#
- list operator: ( ), (,), (=> ,)
- input operator:
  - command input operator ` `
  - line input or angle operator < > (default from @ARGV)
- filename globbing operator * (similar to wild card), and glob( )
- precedence and associativity

precedence	associativity
terms and list operators (leftward)	left
->	left
++ --	nonassociative
**	right
! ~ \ +(unary) –(unary)	right
=~ !~	left
* / % x	left
+ - .	left
<< >>	left
named unary operators	nonassociative
< > <= >= lt gt le ge	nonassociative
== != <=> eq ne cmp	nonassociative
&	left
| ^	left
&&	left
||	left
..	nonassociative
?:	right
= **= += -= .= *= /= %= x= &= |= ^= <<= >>= &&= ||=	right
, =>	left
List operators (rightward)	nonassociative
not	right
and	left
or xor	left

- named unary

-X (file tests)	exists	hex	oct	scalar
alarm	exit	int	ord	sin
caller	exp	lc	quotemeta	sleep
chdir	gethostbyname	lcfirst	rand	sqrt
chroot	getnetbyname	length	readlink	srand
cos	getpgrp	local	ref	stat
defined	getprotobyname	localtime	require	uc
delete	glob	log	reset	ucfirst
do	gmtime	lstat	return	umask
eval	goto	my	mdir	undef

- file test operator

operator	meaning
-r	file is readable by effective uid/gid
-w	file is writable by effective uid/gid
-x	file is executable by effective uid/gid
-o	file is owned by effective uid
-R	file is readable by real uid/gid
-W	file is writable by real uid/gid
-X	file is executable by real uid/gid
-O	file is owned by real uid
-e	file is exists
-z	file has zero size
-s	file has non-zero size (return size)
-f	file is a plain file
-d	file is a directory

-l	file is a symbolic link
-p	file is a named pipe (FIFO)
-S	file is a socket
-b	file is a block special file
-c	file is a character special file
-t	filehandle is opened to a tty
-u	file has setuid bit set
-g	file has setgid bit set
-k	file has sticky bit set
-T	file is a text file
-B	file is a binary file (opposite of –T)
-M	age of file (at startup) in days since modification
-A	age of file (at startup) in days since last access
-C	age of file (at startup) in days since inode change

Control Structures

- Boolean
  - false: undef( ), “”, ‘’, 0, 0.0, “0”, ‘0’, “0.0”, ‘0.0’
  - true:
- conditional: if, elsif, else, unless
- loop: while, until, for, foreach, next, last, undo, <label>:
- error handling: die, warn
- block structure, static local variable “my”, dynamic local variable “local”

Pattern Match with Regular Expression

- pattern matching operator
- pattern binding operator: =~, !~
- substitution operator
- metacharacters: \ | ( ) [{ ^ $ * + ? .
- pattern matching (default: aggressive)
  - exact match
  - whitespace: \s or [\t\n\r\f]
  - non-whitespace: \S
  - digits: \d or [0-9]
  - non-digit: \D
  - word character: \w or [_0-9a-zA-Z]
  - non-word character: \W
- quantifiers
  - fixed repetition: {7}
  - ranges: {7, 11}
  - zero or more: * or {0,}
  - zero or one: ? or {0, 1}
  - one or more: + or {1, }
  - minimum matching: /.*?:/, e.g., /\/*.*?*\// for delete C comments
  - anchor: ^, $, \b, \B, \A (match at beginning of the string), \Z, \G (matches where previous m//g left off)
  - backreference: /<(.*?)>.*?<\/\1>/ for html tags, s/(\S+)\s+(\S+)/$2 $1/ for swap
- modifiers

modifier	meaning
i	do case-insensitive pattern matching
g	match globally, i.e., find all occurrences
m	treat string as multiple lines
s	treat string as single line
e	Evaluate the right side as an expression
o	only compile pattern once
x	extend your pattern’s legibility with whitespace and comments

- modifiers for translate

modifier	meaning
c	Complement the search list
d	Delete found but unreplaced characters
s	Squash duplicate replaced characters

- regular expression extension

(?#text)	a comment
(?: ...)	same as ...
(?= ...)	a zero-width positive lookahead assertion
(?! ...)	a zero-width negative lookahead assertion
(?imsx)	One or more embedded pattern-match modifiers

Format

```
# a report on the /etc/passwd file
Format MY_TOP =
                             Passwd File
Name                         Login   Office  Uid     Gid     Home
--------------------------------------------------------------------------
.
Format MY_FORMAT =
@<<<<<<<<<<<<<< @||||||| @<<< @>>> @>>> @<<<<<
$name                        $login  $office $uid    $gid    $home
.
Select((select(OUTF),            $ofh = select(OUTF);
    $~ = “MY_FORMAT”,            $~ = “MY_FORMAT”;
    $^ = “MY_TOP”                $^ = “MY_TOP”;
)[0]);                           select($ofh);

use FileHandle;
OUTF->format_name(“MY_FORMAT”);
OUTF->format_top_name(“MY_TOP”);

write;
```

Comments

- single-line comment starts with #

- “here” document:

```
print <<EOD;          print <<’EOS’         print `EOC`
    <comments>            <comments>            <commands>
EOD                   EOS                   EOC
```

Functions

- scalar manipulation: chomp, chop, chr, crypt, hex, index, lc, lcfirst, length, oct, ord, pack, q//, qq//, reverse, rindex, sprintf, substr, tr///, uc, ucfirst, y///
- regular expressions and pattern matching: m//, pos, quotemeta, s///, split, study
- numeric functions: abs, atan2, cos, exp, hex, int, log, oct, rand, sin, sqrt, srand
- array processing: pop, push, shift, splice, unshift
- list processing: grep, join, map, qw//, reverse, sort, unpack
- hash processing: delete, each, exists, keys, values
- input and output: binmode, close, closedir, dbmclose, dbmopen, die, eof, fileno, flock, format, getc, print, printf, read, readdir, rewinddir, seek, seekdir, select, syscall, sysread, syswrite, tell, telldir, truncate, warn, write
- fixed-length data and records: pack, read, syscall, sysread, syswrite, unpack, vec
- filehandles, files and directories: chdir, chmod, chown, chroot, fcntl, glob, ioctl, link, lstat, mkdir, open, opendir, readlink, rename, rmdir, stat, symlink, sysopen, umask, unlink, utime
- flow of program control: caller, continue, die, do dump, eval, exit, goto, last, next, redo, return, sub, wantarray
- scoping: caller, import, local, my, package, use
- miscellaneous: defined, dump, eval, formline, local, my, reset, scalar, undef, wantarray
- processes and process groups: alarm, exec, fork, getpgrp, getppid, getpriority, kill, pipe, qx//, setpgrp, setpriority, sleep, system, times, wait, waitpid
- library modules: do, import, no package, require, use
- classes and objects: bless, dbmclose, dbmopen, package, ref, tie, tied, untie, use
- low-level socket access: accept, bind, connect, getpeername, getsockname, getsocketopt, listen, recv, send, setsockopt, shutdown, socket, socketpair
- system V interprocess communication: msgctl, msgget, msgrcv, msgsnd, semctl, semget, semop, shmctl, shmget, shmread, shmwrite
- feching user and group information: endgrent, endhostent, endnetent, endpwent, getgrent, getgrgid, getgrnam, getlogin, getpwent, getpwnam, getpwuid, setgrent, setpwent
- fetching network information: endprotoent, endservent, gethostbyaddr, gethostbyname, gethostent, getnetbyaddr, getnetbyname, getnetent, getprotobyname, getprotobynumber, getprotoent, getservbyname, getservbyport, getservent, sethostent, setnetent, setportoent, setservent
- time: gmtime, localtime, time, times

References/Pointers and Data Structures

- use backslash to get reference, use filehandle as reference, use -> to access attributes, and use $ to dereference
- array of arrays, hash of arrays, array of hashes, hash of hashes, elaborate records, hash of complex records, etc.

Packages, Modules, and Object Classes

- namespaces or packages or module
  - a package is a simple namespace management device
  - a library is a set of subroutines for a particular purpose; a library file with postfix “.pl” and pulled into the main program via “require”, e.g., require Cwd; $here = Cwd::getcwd( );
  - a module is a library that conforms to specific conventions, allowing the file to be brought in with a “use” directive at compile time; the module is the unit of reusability, and ended in “.pm”, e.g, use Cwd; $here = getcwd( );
- a class is simply a packages, and a method is simply a subroutine
- instance variables

```
package HashIntance;                  package ArrayInstance;
sub new {                             sub new {
    my $type = shift;                     my $type = shift;
    my %params = @_;                      my %params = @_;
    my $self = { };                       my $self = [ ];
    $self->{High} = $params{High};        $self->[0] = $params{Left};
    $self->{Low} = $params{Low};          $self->[1] = $params{Right};
    return bless $self, $type;            return bless $self, $type;
}                                     }
package ScalarInstance;
sub new {
    my $type = shift;
    my $self = shift;
    return bless \$self, $type;
}

package main;
$a = HashInstance->new(High =>42, Low => 11);
print “High = $a->{High}\n”;
print “Low = $a->{Low}\n”;
$b = ArrayInstance->new(Left =>78, Right => 40);
print “Left = $b->{Left}\n”;
print “Right = $b->{Right}\n”;
$c = ScalarInstance->new(42);
print “a = $$a\n”;
```

- instance variable inheritance

```
package Base;                         package Derived;
                                      @ISA = qw(Base);
sub new {                             sub new {
    my $type = shift;                   my $type = shift;
    my $self = { };                     my $self = Base->new;
    $self->{buz} = 42;                  $self->{biz} = 11;
    return bless $self, $type;          return bless $self, $type;
}                                   }

package main;
$a = Derived->new;
print "buz = ", $a->{buz}, "\n";
print "biz = ", $a->{biz}, "\n";
```

- containment (the "has-a" relationship)

```
package Inner;                      package Outer;
sub new {                           sub new {
    my $type = shift;                   my $type = shift;
    my $self = { };                     my $self = { };
    $self->{buz} = 42;                  $self->{Inner} = Inner->new;
    return bless $self, $type;          $self->{biz} = 11;
}                                       return bless $self, $type;
                                    }
package main;
$a = Outer->new;
print "buz = ", $a->{Inner}->{buz}, "\n";
print "biz = ", $a->{biz}, "\n";
```

- overriding base class methods

```
package Buz;
sub goo { print "here's the goo\n"; }
package Bar;
@ISA = qw(Buz);
sub google { print "google here\n"; }
package Baz;
sub mumble { print "mumbling\n"; }
package Foo;
@ISA = qw(Bar Baz);
sub new { my $type = shift; return bless [ ], $type; }
sub grr { print "grumble\n"; }
sub goo { my $self = shift; $self->SUPER::goo( ); }
sub mumble { my $self = shift; $self->SUPER::mumble( ); }
sub google { my $self = shift; $self->SUPER::google( ); }

package main;
$foo = Foo->new;
$foo->mumber;
$foo->grr;
$foo->goo;
$foo->google;
```

Cooperating with Other Languages

- program generation: generating other languages in Perl, generating Perl in other languages
- translation from other languages

name	meaning	options
s2p	sed to Perl	-D<num>, -n, -p
a2p	awk to Perl	-D<num>, -F<char>,-n<fieldlist>, -<num>
find2perl	find to Perl	-tar <tarfile>, -eval <string>

- translation to other languages: Perl compiler – perl –MO=C foo.pl>foo.c
- embedding Perl in C and C++
- embedding C and C++ in Perl