附录一 函数集(未定稿)

by flamephoenix

一、进程处理函数

1、进程启动函数

函数名	eval
调用语法	eval(string)
解说	将string看作Perl语句执行。 正确执行后，系统变量$@为空串，如果有错误，$@中为错误信息。
例子	$print = "print (\"hello,world\\n\");"; eval ($print);
结果输出	hello, world

函数名	system
调用语法	system(list)
解说	list中第一个元素为程序名，其余为参数。 system启动一个进程运行程序并等待其结束，程序结束后错误代码左移八位成为返回值。
例子	@proglist = ("echo", "hello,world!"); system(@proglist);
结果输出	hello, world!

函数名	fork
调用语法	procid = fork();
解说	创建程序的两个拷贝--父进程和子进程--同时运行。子进程返回零，父进程返回非零值，此值为子程序的进程ID号。

例子	$retval = fork();
	if ($retval == 0) {
	  # this is the child process
	  exit; # this terminates the child process
	} else {
	  # this is the parent process
	}

结果输出	无

函数名	pipe
调用语法	pipe (infile, outfile);
	与fork合用，给父进程和子进程提供通信的方式。送到outfile文件变量的信息可以通过

解说	infile文件变量读取。步骤： 1、调用pipe 2、用fork将程序分成父进程和子进程 3、一个进程关掉infile，另一个关掉outfile

例子	pipe (INPUT, OUTPUT);
	$retval = fork();
	if ($retval != 0) {
	# this is the parent process
	close (INPUT);
	print ("Enter a line of input:\n");
	$line = <STDIN>;
	print OUTPUT ($line);
	} else {
	# this is the child process
	close (OUTPUT);
	$line = <INPUT>;
	print ($line);
	exit (0);
	}

结果输出	$ program Enter a line of input: Here is a test line Here is a test line $

函数名	exec
调用语法	exec (list);
解说	与system类似，区别是启动新进程前结束当前程序。常与fork合用，当fork分成两个进程后，子进程用exec启动另一个程序。
例子	
结果输出	

函数名	syscall
调用语法	syscall (list);
解说	调用系统函数，list第一个元素是系统调用名，其余为参数。 如果参数是数字，就转化成C的整型数(type int)。否则传递字符串的指针。详见UNIX的帮助或Perl文档。 使用syscall必须包含文件syscall.pl，即： require ("syscall.ph");
例子	
结果输出	

2、进程终止函数

函数名	die
调用语法	die (message);
解说	终止程序并向STDERR输出错误信息。message可以为字符串或列表。如果最后一个参数不包含换行符，则程序文件名和行号也被输出。
例子	die ("Cannot open input file");
结果输出	Cannot open input file at myprog line 6.

函数名	warn

调用语法	warn (message);
解说	与die类似，区别是不终止程序。
例子	warn("Danger! Danger!\n");
结果输出	Danger! Danger!

函数名	exit
调用语法	exit (retcode);
解说	终止程序并指定返回值。
例子	exit(2);
结果输出	无

函数名	kill
调用语法	kill (signal, proclist);
解说	给一组进程发送信号。 signal是发送的数字信号，9为杀掉进程。 proclist是进程ID列表。详见kill的UNIX帮助。
例子	
结果输出	

3、进程控制函数

函数名	sleep
调用语法	sleep (time);
解说	将程序暂停一段时间。time是停止的秒数。返回值为实际停止的秒数。
例子	sleep (5);
结果输出	无

函数名	wait
调用语法	procid = wait();
解说	暂停程序执行，等待子进程终止。 不需要参数，返回值为子进程ID，如果没有子进程，返回-1。
例子	
结果输出	

函数名	waitpid
调用语法	waitpid (procid, waitflag);
解说	暂停程序执行，等待特定的子进程终止。procid为等待的进程ID

例子	$procid = fork();
	if ($procid == 0) {
	  # this is the child process
	  print ("this line is printed first\n");
	  exit(0);
	} else {
	  # this is the parent process
	  waitpid ($procid, 0);
	  print ("this line is printed last\n");
	}

结果输出	$ program this line is printed first this line is printed last $

4、其它控制函数

函数名	caller
调用语法	subinfo = caller();
	返回调用者的程序名和行号，用于Perl Debugger。 返回值为三元素的列表:

解说	1、调用处的包名 2、调用者文件名 3、调用处的行号
例子	
结果输出	

函数名	chroot
调用语法	chroot (dir);
解说	改变程序的根目录，详见chroot帮助。
例子	
结果输出	

函数名	local
调用语法	local($variable);
解说	在语句块(由大括号包围的语句集合)中定义局域变量，仅在此语句块中起作用，对其的改变不对块外同名变量造成影响。 **千万不要在循环中使用，否则每次循环都定义一个新的局域变量！**
例子	
结果输出	

函数名	times
调用语法	timelist = times
解说	返回该程序及所有子进程消耗的工作时间。返回值为四个浮点数的列表: 1、程序耗用的用户时间 2、程序耗用的系统时间 3、子进程耗用的用户时间 4、子进程耗用的系统时间
例子	
结果输出	

二、数学函数

函数名	sin
调用语法	retval = sin (value);
解说	参数为弧度值。

函数名	cos
调用语法	retval = cos (value);
解说	参数为弧度值。

函数名	atan2
调用语法	retval = atan2 (value1, value2);
解说	运算并返回value1除以value2结果的arctan值，单位为弧度，范围在-PI~PI。

应用例:
角度转化成弧度子程序。
	sub degrees_to_radians {
	local ($degrees) = @_;
	local ($radians);11:
	$radians = atan2(1,1) * $degrees / 45;
	}

函数名	sqrt
调用语法	retval = sqrt (value);
解说	平方根函数。value为非负数。

函数名	exp
调用语法	retval = exp (value);
解说	返回e的value次方。

函数名	log
调用语法	retval = log (value);
解说	以e为底的自然对数。

函数名	abs
调用语法	retval = abs (value);
解说	绝对值函数。(Perl 4中没有)

函数名	rand
调用语法	retval = rand (num);
解说	随机数函数，返回0和整数num之间的一个浮点数。

函数名	srand
调用语法	srand (value);
解说	初始化随机数生成器。保证每次调用rand真正随机。

三、字符串处理函数

函数名	index
调用语法	position = index (string, substring, position);
解说	返回子串substring在字符串string中的位置，如果不存在则返回-1。参数position是可选项，表示匹配之前跳过的字符数，或者说从该位置开始匹配。

函数名	rindex
调用语法	position = rindex (string, substring, position);
解说	与index类似，区别是从右端匹配。

函数名	length
调用语法	num = length (string);
解说	返回字符串长度，或者说含有字符的数目。

函数名	pos
调用语法	offset = pos(string);
解说	返回最后一次模式匹配的位置。

函数名	substr
调用语法	substr (expr, skipchars, length)
解说	抽取字符串（或表达式生成的字符串）expr中的子串，跳过skipchars个字符，或者说从位置skipchars开始抽取子串（第一个字符位置为0），子串长度为length，此参数可忽略，意味着取剩下的全部字符。 当此函数出现在等式左边时，expr必须为变量或数组元素，此时其中部分子串被等式右边的值替换。

函数名	study
调用语法	study (scalar);
解说	用一种内部格式提高变量的访问速度，同一时刻只对一个变量起作用。

函数名	lc uc

调用语法	retval = lc(string); retval = uc(string);
解说	将字符串全部转换成小/大写字母。

函数名	lcfirst ucfirst
调用语法	retval = lcfirst(string); retval = ucfirst(string);
解说	将第一个字母转换成小/大写。

函数名	quotameta
调用语法	newstring = quotemeta(oldstring);
解说	将非单词的字母前面加上反斜线(\)。 语句： $string = quotemeta($string); 等效于: $string =~ s/(\W)/\\$1/g; 常用于模式匹配操作中，确保字符串中没有字符被看作匹配操作符。

函数名	join
调用语法	join (joinstr, list);
解说	把字符串列表(数组)组合成一个长的字符串，在每两个列表元素间插入串joinstr。

函数名	sprintf
调用语法	sprintf (string, fields);
解说	与printf类似，区别是结果不输出到文件，而作为返回值赋给变量。
例子	$num = 26; $outstr = sprintf("%d = %x hexadecimal or %o octal\n",$num, $num, $num); print ($outstr);
结果输出	26 = 1a hexadecimal or 32 octal

四、标量转换函数

函数名	chop
调用语法	$lastchar = chop (var);
解说	var可为变量或数组，当var为变量时，最后一个字符被删除并赋给$lastchar，当var为数组/列表时，所有元素的最后一个字符被删除，最后一个元素的最后一个字母赋给$lastchar。

函数名	chomp
调用语法	result = chomp(var);
解说	检查字符串或字符串列表中元素的最后一个字符是否为由系统变量$/定义的行分隔符，如果是就删除。返回值为实际删除的字符个数。

函数名	crypt
调用语法	result = crypt (original, salt);
解说	用DES算法加密字符串，original是将要加密的字符串，salt是两个字符的字符串，定义如何改变DES算法，以使更难解码。返回值为加密后的串。

函数名	hex
调用语法	decnum = hex (hexnum);
解说	将十六进制数(字符串形式)转化为十进制数。

函数名	int
调用语法	intnum = int (floatnum);

解说	将浮点数舍去小数部分转化为整型数。

函数名	oct
调用语法	decnum = oct (octnum);
解说	将八进制数(字符串形式)或十六进制数("0x.."形式)转化为十进制数。

函数名	ord
调用语法	asciival = ord (char);
解说	返回单个字符的ASCII值，与PASCAL中同名函数类似。

函数名	chr
调用语法	$char = chr (asciival);
解说	返回ASCII值的相应字符，与PASCAL中同名函数类似。

函数名	pack
调用语法	formatstr = pack(packformat, list);
解说	把一个列表或数组以在实际机器存贮格式或C等编程语言使用的格式转化（包装）到一个简单变量中。参数packformat包含一个或多个格式字符，列表中每个元素对应一个，各格式字符间可用空格或tab隔开，因为pack忽略空格。 除了格式a、A和@外，重复使用一种格式多次可在其后加个整数，如: $twoints = pack ("i2", 103, 241); 把同一格式应用于所有的元素则加个*号，如: $manyints = pack ("i*", 14, 26, 11, 83); 对于a和A而言，其后的整数表示要创建的字符串长度，重复方法如下: $strings = pack ("a6" x 2, "test1", "test2"); 格式@的情况比较特殊，其后必须加个整数，该数表示字符串必须的长度，如果长度不够，则用空字符(null)补足，如: $output = pack ("a @6 a", "test", "test2"); pack函数最常见的用途是创建可与C程序交互的数据，例如C语言中字符串均以空字符(null)结尾，创建这样的数据可以这样做: $Cstring = pack ("ax", $mystring); 下表是一些格式字符与C中数据类型的等价关系:

字符	等价C数据类型
C	char
d	double
f	float
i	int
I	unsigned int (or unsigned)
l	long
L	unsigned long
s	short
S	unsigned short

完整的格式字符见下表。

格式字符	描述
a	用空字符(null)补足的字符串
A	用空格补足的字符串
b	位串，低位在前
B	位串，高位在前
c	带符号字符（通常-128~127）
C	无符号字符（通常8位）
d	双精度浮点数
f	单精度浮点数

h	十六进制数串，低位在前
H	十六进制数串，高位在前
i	带符号整数
I	无符号整数
l	带符号长整数
L	无符号长整数
n	网络序短整数
N	网络序长整数
p	字符串指针
s	带符号短整数
S	无符号短整数
u	转化成uuencode格式
v	VAX序短整数
V	VAX序长整数
x	一个空字节
X	回退一个字节
@	以空字节(null)填充

函数名	unpack
调用语法	@list = unpack (packformat, formatstr);
解说	unpack与pack功能相反，将以机器格式存贮的值转化成Perl中值的列表。其格式字符与pack基本相同（即上表），不同的有：A格式将机器格式字符串转化为Perl字符串并去掉尾部所有空格或空字符；x为跳过一个字节；@为跳过一些字节到指定的位置，如@4为跳过4个字节。下面看一个@和X合同的例子：　$longrightint = unpack ("@* X4 L", $packstring); 此语句将最后四个字节看作无符号长整数进行转化。下面看一个对uuencode文件解码的例子： 1 : #!/usr/local/bin/perl 2 : 3 : open (CODEDFILE, "/u/janedoe/codefile") \|\| 4 : die ("Can't open input file"); 5 : open (OUTFILE, ">outfile") \|\| 6 : die ("Can't open output file"); 7 : while ($line = <CODEDFILE>) { 8 : $decoded = unpack("u", $line); 9 : print OUTFILE ($decoded); 10: } 11: close (OUTFILE); 12: close (CODEDFILE); 当将pack和unpack用于uuencode时，要记住，虽然它们与UNIX中的uuencode、uudecode工具算法相同，但并不提供首行和末行，如果想用uudecode对由pack的输出创建的文件进行解码，必须也把首行和末行输出（详见UNIX中uuencode帮助）。

函数名	vec
调用语法	retval = vec (vector, index, bits);
解说	顾名思义，vec即矢量(vector)函数，它把简单变量vector的值看作多块(维)数据，每块含一定数目的位，合起来即一个矢量数据。每次的调用访问其中一块数据，可以读取，也可以写入。参数index就象数组下标一样，提出访问哪一块，0为第一块，依次类推，要注意的是访问次序是从右到左的，即第一块在最右边。参数bits指定每块中的位数，可以为1,2,4,8,16或32。
例子	1 : #!/usr/local/bin/perl 2 : 3 : $vector = pack ("B*", "11010011"); 4 : $val1 = vec ($vector, 0, 4); 5 : $val2 = vec ($vector, 1, 4); 6 : print ("high-to-low order values: $val1 and $val2\n"); 7 : $vector = pack ("b*", "11010011");

	8 : $val1 = vec ($vector, 0, 4); 9 : $val2 = vec ($vector, 1, 4); 10: print ("low-to-high order values: $val1 and $val2\n");
结果	high-to-low order values: 3 and 13 low-to-high order values: 11 and 12

函数名	defined
调用语法	retval = defined (expr);
解说	判断一个变量、数组或数组的一个元素是否已经被赋值。expr为变量名、数组名或一个数组元素。 如果已定义，返回真，否则返回假。

函数名	undef
调用语法	retval = undef (expr);
解说	取消变量、数组或数组元素甚至子程序的定义，回收其空间。返回值始终为未定义值，此值与空串等效。

五、数组和列表函数

函数名	grep
调用语法	@foundlist = grep (pattern, @searchlist);
解说	与同名的UNIX查找工具类似，grep函数在列表中抽取与指定模式匹配的元素，参数pattern为欲查找的模式，返回值是匹配元素的列表。
例子	@list = ("This", "is", "a", "test"); @foundlist = grep(/^[tT]/, @list);
结果	@foundlist = ("This", "test");

函数名	splice
调用语法	@retval = splice (@array, slipelements, length, @newlist);
解说	拼接函数可以向列表（数组）中间插入元素、删除子列表或替换子列表。参数skipelements是拼接前跳过的元素数目，length是被替换的元素数，newlist是将要拼接进来的列表。当newlist的长度大于length时，后面的元素自动后移，反之则向前缩进。因此，当length=0时，就相当于向列表中插入元素，而形如语句 splice (@array, -1, 0, "Hello"); 则向数组末尾添加元素。而当newlist为空时就相当于删除子列表，这时，如果length为空，就从第skipelements个元素后全部删除，而删除最后一个元素则为：splice (@array, -1);这种情况下，返回值为被删去的元素列表。

函数名	shift
调用语法	element = shift (@arrayvar);
解说	删去数组第一个元素，剩下元素前移，返回被删去的元素。不加参数时，缺省地对@ARGV进行操作。

函数名	unshift
调用语法	count = unshift (@arrayver, elements);
解说	作用与shift相反，在数组arrayvar开头增加一个或多个元素，返回值为结果(列表)的长度。等价于splice (@array, 0, 0, elements);

函数名	push
调用语法	push (@arrayvar, elements);
解说	在数组末尾增加一个或多个元素。等价于slice (@array, @array, 0, elements);

函数名	pop
调用语法	element = pop (@arrayvar);
解说	与push作用相反，删去列表最后一个元素，并将其作为返回值，当列表已空，则返回“未定义值”(即空串)。

函数名	split
调用语法	@list = split (pattern, string, maxlength);
解说	将字符串分割成一组元素的列表。每匹配一次pattern，就开始一个新元素，但pattern本身不包含在元素中。maxlength是可选项，当指定它时，达到该长度就不再分割。

函数名	sort
调用语法	@sorted = sort (@list);
解说	按字母次序给列表排序。

函数名	reverse
调用语法	@reversed = reverse (@list);
解说	按字母反序给列表排序。

函数名	map
调用语法	@resultlist = map (expr, @list);
解说	此函数在Perl5中定义，可以把列表中的各个元素作为表达式expr的操作数进行运算，其本身不改变，结果作为返回值。在表达式expr中，系统变量$_代表各个元素。
例子	1、@list = (100, 200, 300); @results = map ($_+1, @list); 2、@results = map (&mysub($_), @list);
结果	1、(101, 201, 301) 2、无

函数名	wantarray
调用语法	result = wantarray();
解说	Perl中，一些内置函数的行为根据其处理简单变量还是数组有所不同，如chop。自定义的子程序也可以定义这样两种行为。当子程序被期望返回列表时，此函数返回值为非零值(真)，否则为零值(假)。

例子	1 : #!/usr/local/bin/perl
	2 :
	3 : @array = &mysub();
	4 : $scalar = &mysub();
	5 :
	6 : sub mysub {
	7 : if (wantarray()) {
	8 : print ("true\n");
	9 : } else {
	10: print ("false\n");
	11: }
	12: }

结果	$program true false $

六、关联数组函数

函数名	keys

调用语法	@list = keys (%assoc_array);
解说	返回关联数组无序的下标列表。

函数名	values
调用语法	@list = values (%assoc_array);
解说	返回关联数组无序的值列表。

函数名	each
调用语法	@pair = each (%assoc_array);
解说	返回两个元素的列表--键值对（即下标和相应的值），同样无序。当关联数组已空，则返回空列表。

函数名	delete
调用语法	element = delete (assoc_array_item);
解说	删除关联数组中的元素，并将其值作为返回值。
例子	%array = ("foo", 26, "bar", 17"); $retval = delete ($array{"foo"});
结果	$retval = 26;

函数名	exists
调用语法	result = exists (element);
解说	在Perl5中定义，判断关联数组中是否存在某元素，若存在，返回非零值(真)，否则返回零值(假)。
例子	$result = exists ($myarray{$mykey});