

Perl Operators

The following table lists Perl's operators - those symbols that allow you to perform some sort of action on program data:

<i>Operator</i>	<i>Description</i>
X	Character repeat and list repeat.
.	String concatenation.
=	Assignment.
+	Add.
-	Subtract.
*	Multiply.
/	Divide.
%	Modulo divide.
-	Unary negation: negates the number it precedes.
+	Unary plus: does nothing.
^	Raise to the power.
++	Unary increment, can be used before or after a variable to increment before or after the variable is evaluated. Therefore, if \$ a is 5, \$ a++ is 5 until it is evaluated, and then \$a becomes 6; ++\$a is 6 when it's evaluated.
--	Unary decrement; can be used before or after a variable to decrement before or after the variable is evaluated. Thus, if \$a is 5, \$a -- is 5 until it's evaluated, and then \$ a becomes 4; -- \$ a is 4 when it's evaluated.
+=, = etc.	Assignment after operation. For example, \$a += 3 means "add 3 to \$a and store that value in \$ a."
,	List separator.
..	Range.

< > <= >= == != < = >	Numeric comparisons: less than, greater than, less than or equal to, greater than or equal to, equal to, not equal to, comparison.
lt gt le ge eq ne cmp	String comparisons: less than, greater than, less than or equal to, greater than or equal to, equal to, not equal to, comparison.
not	Logical not. Can also be !.
and	Logical and. Can also be &&.
or	Logical or. Can also be
xor	Logical xor.
?:	Conditional comparison: TEST ? IFTRUE : IFFALSE.
~	Bitwise logical not.
&	Bitwise logical and.
	Bitwise logical or.
^	Bitwise logical xor.
<< >>	Bit shift to the left and to the right.
< >	1/0 operator, such as < STDIN >
=~ !~	Binds a scalar on the left to a pattern match, substitution, or translation on the right. returns the inverse of the bound value.
m/PATTERN/MODIFIERS	Searches for the pattern and returns true if found. Can <u>also be specified</u> without the m.
s/PATTERN/REPLACEMENT/MODIFIERS	Searches for the pattern and replaces it with the specified text.
tr/FROMRANGE/TORANGE/	Returns a string with letters translated from one range to another. The t r and y operators are identical.
Operator	Description
= >	List separator; same as a comma.
` (backtick)	Runs the command between the two backticks and returns the STDOUT.
- >	Dereference. Usually used with object-oriented programs as CLASS- >METHOD.
\	Reference.

Perl's Operator Hierarchy

Although using parentheses to group your operations is always safer, some programmers prefer to trust Perl's rules for deciding which operators are evaluated first. The following table lists the operators in order of their precedence. Those operators with the same ranking (they're shown on the same line in the list that follows), Perl evaluates them from left to right.

For instance, the following statement

```
$a = 3 / 4 ** 2;
```

is interpreted as

```
$a = 3 / (4 ** 2);
```

because ****** has higher precedence than **/** and is, therefore, evaluated first.

Operators are evaluated in the following order (lower numbers mean higher precedence):

1. - >
2. ++ --
3. **
4. ! ~ \ [unary +1] [unary -1]
5. =~ !~
6. * / % % x
7. + - .
8. << >>
9. < > <= >= lt gt le ge
10. == <= > eq ne cmp
11. &
12. |
13. &&
14. | |
15. .
16. ?:
17. += and other operators that end in
18. =>
19. not
20. and
21. or xor

Perl's Functions and Statements

The following table lists all of Perl's functions and statements in alphabetical order:

Name	Description
abs(NUMBER)	Takes the absolute value of the argument (that is, negative numbers are made positive).
accept(NEWSOCKETHANDLE, GENERICSOCKETHANDLE)	Accepts a connection on a socket.

alarm(TIME)	Sends the SIGALRM signal to the program after a specified number of seconds.
atan2(Y, X)	Returns the arctangent of Y/X.
bind(SOCKET, NAME)	Assigns a name to an existing open socket
binmode(FILEHANDLE)	Tells Perl to treat the file as a binary file. It has no function under UNIX or MacOS.
bless(REFERENCE, PACKAGE)	Tells the item that is referred to in the reference that it is an object in the specified package. If the package is not specified, Perl uses the current package.
caller(NUMBEROFSTACKS)	Provides information on the stack of subroutine calls. The argument is the number of subroutine calls to unnest. With an argument, it returns a list with the package name, filename, line number, subroutine name, whether the subroutine has arguments, and whether the subroutine wants an array as its argument. Without an argument, caller returns just the first three list items, referring to the current subroutine.
chdir(DIRNAME)	Changes the working directory of a program to a specified directory or, with no argument, changes the directory to the home directory of the user. Returns true if successful, false if it isn't (for example, if the directory doesn't exist).
chmod(NUMMODE, LISTOFFILES)	Changes the access permissions on one or more files. The first argument must be an octal file mode, such as 0777. The function returns the number of files successfully changed.
chomp(STRINGVAR)	Removes the line ending from the end of a string variable, returning the number of characters deleted. If the last character is not a line-ending character, as defined in the special variable \$/, nothing is removed. If \$/ is two characters, both characters are removed. Note: This function changes the value of the variable. You can also use the function chomp (LIST), which removes the line ending from the end of each list item.
chop(STRINGVAR)	Removes the last character from the end of the string variable, returning the character deleted. Note: This function changes the value of the variable.
chown(USERID, GROUPID,	Changes the user and group ownership of one or

<code>LISTOFFILES)</code>	more files. The <code>userid</code> and <code>groupid</code> must be numbers, not names. The function returns the number of files successfully changed.
<code>chr(NUMBER)</code>	Returns the character represented by the argument.
<code>chroot(DIRECTORY)</code>	Changes the root directory available to a program to the specified directory. The named directory becomes the root directory to the program, and the program cannot modify or see other directories above this one in the directory hierarchy.
<code>close (FILEHANDLE</code>	Closes a file that was opened with the <code>open</code> function.
<code>closedir(DIRHANDLE)</code>	Closes a directory that was opened with the <code>opendir</code> function.
<code>connect(SOCKET, NETADDRESS)</code>	Connects to another process using the specified socket.
<code>continue { BLOCK }</code>	Executes the block. The <code>continue</code> statement always follows other blocks because it is executed when other blocks exit.
<code>cos(NUMBER)</code>	Returns the cosine of the number, expressed in radians.
<code>crypt(PLAINTEXT, SALT)</code>	Encrypts the first argument using the <code>salt</code> parameter. This function is system dependent, so you may get varying results, depending on the operating system.
<code>dbmclose(ARRAY)</code>	Breaks the binding between the array and the DBM file that was associated with the array by an earlier <code>dbmopen</code> function.
<code>dbmopen(ARRAY, FILENAME, MODE)</code>	Begins the process of associating an associative array with the named database. The mode is the octal file mode used to open the database. This function clears out any values already in the associative array.
<code>defined(VARIABLE)</code>	Returns true if the variable has been defined (that is, if it has a valid string, numeric, or reference value). This function is useful for testing undefined values returned under error conditions.

<code>delete(\$ARRAY(KEYI))</code>	Removes a key and its value from an associative array.
<code>die(TEXT)</code>	Prints the text to the standard error output and terminates the Perl program. The program returns the value that was in the <code>\$</code> special variable.
<code>do { BLOCK }</code>	Executes the block and returns the value of the last expression in the block.
<code>do(FILENAME)</code>	Executes the statements in the file and returns the value of the last expression in the file.
<code>dump</code>	Causes a core dump, if the operating system supports it. You can also specify a label, which the program will start from if you can recreate the core from the dump.
<code>each(ARRAY)</code>	Returns a two-item list (the key and the value) for the next value in the array.
<code>endgrent</code>	Resets the cycling through the group list begun by the <code>getgrent</code> function.
<code>endhostent</code>	Resets the cycling through the host list begun by the <code>gethostent</code> function.
<code>endnetent</code>	Resets the cycling through the network list begun by the <code>getnetent</code> function.
<code>endprotoent</code>	Resets the cycling through the protocol list begun by the <code>getprotoent</code> function.
<code>endpwent</code>	Resets the cycling through the password list begun by the <code>getpwent</code> function.
<code>endservent</code>	Resets the cycling through the server list begun by the <code>getservent</code> function.
<code>eof(FILEHANDLE)</code>	Returns true if the next read on the file handle would read off the end of the file.
<code>eval(EXPRESSION)</code>	Causes Perl to execute the expression as though the expression were a Perl program. The expression can be one or more statements.
<code>exec(PROGRAM)</code>	Causes the Perl program to stop and the program named in the argument to be run. The argument can be a list, in

	which case each item is passed as an element of the command line to be executed.
exists(ARRAY(\$KEY))	Returns true if the specified key exists in the specified array.
exit(NUMBER)	Stops the program. The argument is used as the return value for the program.
exp(NUMBER)	Returns the number e raised to the power of the argument.
fcntl(FILEHANDLE, FUNCTION, SCALAR)	Executes the UNIX fcntl function for performing low-level file control.
File tests. -X FILENAME	Returns values based on the file. File tests include determining whether the file exists and how many bytes are in the file, and other tests.
fileno(FILEHANDLE)	Returns the numeric file descriptor of the handle given, or the undef value if the file is not open.
flock(FILEHANDLE OPERATION)	Performs low-level file locking in UNIX.
for (INITEXPR; TESTEXPR; ENDEXPR) { BLOCK }	Executes the block repeatedly. The initial expression is evaluated first, followed by the test expression. If the test expression is true, the block is executed, and the end expression is evaluated. The test expression is then evaluated, and the loop continues.
foreach VARIABLE (LIST) { BLOCK }	Executes the block repeatedly. For each item in the list, the variable is set to the item's value and the block is executed.
fork	Starts a child process and returns its process ID number.
format NAME= FORMATLIST	Creates a format that can be used by the write function. The format list consists of lines that contain picture formats, lists of arguments, or comments.
formline (PICTURE, LIST)	Formats a list of values according to the picture.

getc(FILEHANDLE)	Returns the next byte from the file, or a null string if you're at the end of the file. If called without an argument, it returns the next byte from the standard file input.
getgrent	Returns UNIX group information from the next line of the /etc/group file. The list returned has four items: the group name, the group pass word (encrypted), the group number, and a string of all the members' names.
getgrgid(GROUPID)	Returns information about the group, based on the group ID number. The list returned has four items: the group name, the group pass word (encrypted), the group number, and a string of all the members' names.
getgrnam(GROUPNAME)	Returns information about the group, based on the group's name. The list returned has four items: the group name, the group pass word (encrypted), the group number, and a string of all the members' names.
gethostbyaddr (PACKEDADDR, ADDRTYPE)	Returns information about the host based on its address. The address argument is packed, probably with the pack('C4') function, and the address type is always 2 for Internet addresses. In scalar context, gethostbyaddr returns the host name; in list context, it returns the host name, a string of aliases, the address type, the length of the returned list of packed addresses, and a list of packed addresses.
gethostbyname (DOMAINNAME)	Returns information about the host based on its domain name. In scalar context, gethostbyname returns the IP address, packed; in list context, it returns the host name, a string of aliases, the address type, the length of the returned list of packed addresses, and a list of packed addresses. You can unpack the IP address with the unpack('C4') function.
gethostent	Returns the next record from the /etc/ hosts file. It returns the host name, a string of aliases, the address type, the length of the returned list of packed addresses, and a list of packed addresses.

getlogin	Returns the current login name by querying the /etc/utmp file.
getnetbyaddr(PACKEDADDR, ADDRTYPE)	Returns information about the network based on its address from the /etc/networks file. The address argument is packed, probably with the pack('C4') function, and the address type is always 2 for Internet addresses. In scalar context, getnetbyaddr returns the network name; in list context, it returns the network name, a string of aliases, the address type, and the packed address.
getnetbyname(DOMAINNAME)	Returns information about the network based on its name from the /etc/networks file. In scalar context, getnetbyname returns the network name; in list context, it returns the network name, a string of aliases, the address type, and the packed address.
getnetent	Returns information from the next line in the /etc/networks file. In scalar context, getnetent returns the network name; in list context, it returns the network name, a string of aliases, the address type, and the packed address.
getpeername(SOCKET)	Returns the socket address of the other end of a connection. The address returned is packed.
getpgrp(PROCESSID)	Returns the process group number for the given process number. Use an argument of 0 to indicate the current process.
getppid	Returns the process number of the parent process.
getpriority (PROCESSKIND, WHO)	Returns the priority of a process.
getprotobyname (PROTONAME)	Returns information about a protocol based on its name. In scalar context, the protocol number is returned. In list context, the items in the list returned are the protocol name, aliases for the name, and the protocol number.
getprotobynumber (PROTONUMBER)	Returns information about a protocol based on its number. In scalar context, the protocol name is returned. In list context, the

	items in the list returned are the protocol name, aliases for the name, and the protocol number.
getprotoent	Returns the next record from the <code>/etc/protocols</code> file. In scalar context, the protocol name is returned. In list context, the items in the list returned are the protocol name, aliases for the name, and the protocol number.
getpwent	Returns user information from the next line of the <code>/etc/passwd</code> file. The list returned has nine items: the user name, the password (encrypted), the user number, the group number, the quota field, the comment field, the gcos field, the home directory, and the default shell. In scalar context only the user name is returned.
getpwnam(USERNAME)	Returns user information for the specified user name. The list returned has nine items: the user name, the password (encrypted), the user number, the group number, the quota field, the comment field, the gcos field, the home directory, and the default shell. In scalar context, only the user number is returned.
getpwuid(USERID)	Returns user information for the specified user number. The list returned has nine items: the user name, the password (encrypted), the user number, the group number, the quota field, the comment field, the gcos field, the home directory, and the default shell. In scalar context, only the user name is returned.
getservbyname (SERVICENAME, PROTO)	Returns information about a service based on its name. In scalar context, the port number is returned. In list context, the items in the list returned are the service name, aliases for the name, port number, and protocol name.
getservbyport (PORT, PROTO)	Returns information about a service based on its port number. In scalar context, the service name is returned. In list context, the items in the list returned are the service name, aliases for the name, port number, and protocol name.
getservent	Returns information from the next record in the <code>/etc/services</code> file. In scalar context, the service name is returned. In list context, the items in the list returned are the service

	name, aliases for the name, the port number, and protocol name.
getsockname(SOCKET)	Returns the socket address of the local end of a connection. The address returned is packed.
getsockopt(SOCKET, LEVEL, OPTIONNAME)	Returns the socket option specified.
glob(STRING)	Returns a list of files matching the argument, the argument may have wildcard characters.
gintime(TIME)	Converts the given time into Greenwich Mean Time. In scalar context, this function returns a string that looks something like Sat Nov 23 20:07:47 1996 (in Perl5 and later versions only); in list context the items are: second, minute, hour, day of month, month (January = 0, February = 1,...), year, weekday (Sunday = 0, Monday= 1,...), day of the year (January 1 = 0, January 2 = 1, . . . and standard time (true or false). With no argument, the function assumes the current time.
goto LABEL	Jumps to the named label. You can use an expression instead of a label name, and Perl interprets that expression into a label name.
grep(BLOCK, LIST)	Evaluates the block in Boolean context on each item of the list. In scalar context, it returns the number of times that the block evaluated true; in list context, it returns the list of items matched.
grep(REGEXP, LIST)	Searches for the regular expression in each item of the list. In scalar context, returns the number of times that the regular expression was found; in list context, returns the list of items matched.
hex(STRING)	Returns the number that interprets the string as hexadecimal digits.
if(TEST) { BLOCK1 }	Executes the block if the test is true. The first block may be followed by else { BLOCK2 } In this case, if the test is false, the second block is executed. The first block may also be followed by one or more instances of e l s i f

	{ BLOCKn }, which acts the same as else if.
import(CLASSNAME)	This class method exports the class name to the current module.
index(STRING, SUBSTRING, POSITION)	Returns the position of the first occurrence of a substring in the string, or -1 if the substring isn't found. If the position is specified, Perl starts looking from that position instead of from the beginning of the string.
int(NUMBER)	Returns the integer portion of the number.
ioctl(FILEHANDLE, FUNCTION, SCALAR)	Executes UNIX's ioctl function for performing low-level input and output.
join(SEPARATOR, LIST)	Returns a string that consists of the items in the list with the separator between each item. The separator is not added to the beginning or end of the resulting string.
keys(ARRAY)	Normally used to return a list of all the keys in the associative array. In scalar context, it returns the number of elements in the associative array.
kill(SIGNAL, LIST)	Sends the signal to the list of process IDs. The signal is either the signal number or the name of the signal, for example, HUP. The function returns the number of processes signaled.
last	Exits the innermost loop that is executing. You can also give an argument of a label, in which case Perl exits out to the loop with that label.
lc(STRING)	Returns the string in all lowercase characters.
lcfirst(STRING)	Returns the string with the first character changed to lowercase.
length(STRING)	Returns the length of the string.
link(OLDFILE, NEWFILE)	Creates a new file that has a hard link to the old file. Returns true if successful.
listen(SOCKET, MAXCONN)	Causes the system to allow connections on the socket. The system queues up to the number of connections specified.

local(VARIABLES)	Makes one or more variables local to the current block, subroutine, e v a l function, or file.
localtime(TIME)	Converts the given time into the local time. In scalar context, this returns a string that looks something like Sat Nov 23 20:07:47 1996 (in Perl 5 only); in list context the items are: second, minute, hour, day of month, month (January = 0, February=1...), year, weekday (Sunday = 0, Monday = 1,...), day of the year (January 1 = 0, January 2 = 1,...), and standard time (true or false). With no argument, the function assumes the current time.
log(NUMBER)	Returns the natural logarithm (base e) of the number.
lstat(FILE)	Returns information about the file. The argument can be either a file handle or a string with the file's name. If the file is a symbolic link, the information is about the linked-to file, not the link itself. The following items in the list are returned: the device number of the filesystem, inode number, file permissions (number), number of hard links to the file, user number of file's owner, group number of the file's owner, device identifier for special files, size of file, last access time, last modify time, last inode change time, preferred block size, and number of blocks allocated.
map BLOCK LIST	Returns a list of the block evaluated in list context for each item in the list. During evaluation, \$_ is set to the list item being evaluated.
map(EXPRESSION, LIST)	Returns a list of the expression evaluated in list context for each item in the list. During evaluation, \$_ is set to the list item being evaluated.
mkdir(DIRNAME, PERMISSION)	Creates a directory with the specified permissions (in octal). Returns true if successful.
msgctl(ID, COMMAND, ARGUMENT)	Sends a message using UNIX's msgctl function.
msgget(KEY, FLAGS)	Returns the message ID for System V IPC messages.
msgrcv(ID, VARIABLE, SIZE, TYPE, FLAGS)	Receives a message with the message ID.

<code>msgsnd(ID, MESSAGE, FLAGS)</code>	Sends a message with the message ID.
<code>my(VARIABLES)</code>	Makes one or more variables local to the current block, subroutine, ev a 1 function, or file.
<code>new(CLASSNAME)</code>	Constructs an object from the class. You can also specify a list after the class name as arguments passed to the constructor.
<code>next</code>	Causes the innermost loop to start again from the beginning of the loop. You can also include a label with the function, in which case Perl jumps to the beginning of the loop with that label.
<code>no MODULE</code>	Removes subroutine and variable names that were imported by the use function after the use function is called on the same module name.
<code>oct(STRING)</code>	Returns the number that is the string interpreted as an octal value.
<code>open(FILEHANDLE, NAME)</code>	Opens the named file or command and associates it with the given file handle. The characters at the beginning of the name determine whether the file is opened for input, output, or appending. In the case of a command, the location of the character in the command name determines whether the file handle is for reading or writing. The function returns true if the file or command is opened successfully.
<code>opendir(DIRHANDLE, DIRNAME)</code>	Opens the directory for reading and associates the directory handle with that directory.
<code>ord(STRING)</code>	Returns the ASCII value of the first character of the string.
<code>our(VARIABLES)</code>	Makes one or more variables local to the current block, subroutine, e v a 1 function, or file. Unlike my, o u r doesn't create a local variable.
<code>pack(TEMPLATE, LIST)</code>	Returns a string containing the items from the list put together using the template.
<code>package NAME</code>	Declares that the rest of the innermost block, subroutine, e v a l function, or file belongs to

	the specified package name.
pipe(READHANDLE, WRITEHANDLE)	Opens a pair of pipes for use by other functions.
pop(LISTNAME)	Shortens the list by removing the last item, and returns that item. If the list is empty, it returns the undefined value. Note.-The argument must be the name of a list, not an actual list.
pos(VARIABLE)	Returns the position in a variable where the last m/ / g associated with the variable left off.
print(FILEHANDLE, LIST)	Sends a list to the file specified by the file handle and returns true if successful. Note that the argument is a list, not a scalar; therefore, variables that are arguments to print are evaluated in list context. The print function is more commonly written without parentheses.
print(LIST)	Sends the list to the standard file output and returns true if successful. The argument is a list,-not a scalar; therefore, variables that are arguments to print are evaluated in list context. The print function is more commonly written without parentheses.
printf(FILEHANDLE FORMAT, LIST)	Formats a list using the format string, sends the resulting list to the file specified by the filehandle, and returns true if successful. <i>Note.</i> No comma appears after the file handle.
printf (FORMAT, LIST)	Formats a list using the format string, sends the resulting list to the standard file output, and returns true if successful.
push(LISTNAME, LIST)	Adds the items of the list in the second argument to the end of the list named in the first argument.
q/STRING/	Returns a string with no interpretation of the string's contents. This function is similar to using single quotes (' '), except that you can choose the quoting character.
qq/STRING/	(luotes a string, with interpretation of the string's contents. This function is similar to using double quotes C' "), except that you can choose the quoting character.
quotemeta(STRING)	Returns the value of an argument with all the

	regular expression metacharacters backslashed as necessary.
qw/STRING/	Returns the words in a string with no interpretation. Ignores differences in white space between words.
qx/COMMAND/	Returns the output of running a command. This function is similar to using the backtick character (`), except that you can choose the quoting character.
rand(NUMBER)	Returns a random number between 0 and the specified number (including 0 but not including the specified number). If no argument is given, it returns a random number between 0 and 1. The seed used for the random number is set with the s r a n d function.
read(FILEHANDLE, VARIABLE, LENGTH, OFFSET)	Reads the number of bytes from the file into the scalar variable. The function starts writing into the variable at the given offset, which is optional. The function returns the actual number of bytes read or 0 if you are already at the end of the file; the actual number may be less than the desired number if the function read off the end of the file.
readdir(DIRHANDLE)	Reads the next entry from a directory. In scalar context, the function returns the next filename in the directory; in list context, the function returns the rest of the filenames.
readlink(FILENAME)	Returns the name of the file pointed to by the symbolic link in the filename given in the argument. If the named file is not a symbolic link or some other error occurs, the function returns the undefined value.
recv(SOCKET, VARIABLE, LENGTH, FLAGS)	Receives a message on the socket and puts it in the variable.
redo	Restarts the innermost loop without reevaluating the conditional test for the loop. You can also give an argument of a label, which causes Perl to restart the loop with that label.
ref(VARIABLE)	Returns true if the variable is a reference, otherwise returns the null string. The value

	returned is a string indicating the type of reference.
rename(FILENAME, NEWNAME)	Renames a file and returns true if successful. This function does not work across file systems on UNIX.
require (FILENAME)	Executes the named Perl program provided that program has not already been executed.
require(NUMBER)	Stops the program if the Perl version number is lower than the argument.
require(PACKAGENAME)	Loads the package in the same way the use function does, except that the package is loaded while the program is running, not when it is compiled.
reset	Resets variables and single-match searches performed with the ?? operator.
reset(STRING)	Resets all variables that have a name starting with the letter in the string. If the string is more than one character, that string must be a range expressed with a hyphen, in which case all variables whose first letter starts with any letter in the given range are reset. Avoid using this function.
return(LIST)	Returns the specified value from a subroutine. If the subroutine is in scalar context, the first item in the list is returned; if the subroutine is in list context, the entire list is returned.
reverse(LIST)	Returns a list in the reverse order of the argument.
rewinddir(DIRHANDLE)	Sets the position of the next readdir function to the beginning of the directory.
rindex(STRING, SUBSTRING, POSITION)	Returns the position of the last occurrence of the substring in the string, or -1 if the substring isn't found. If the position is specified, Perl starts looking from that position instead of from the beginning of the string.
rmdir(DIRECTORYNAME)	Removes the directory provided the directory is empty.

scalar(EXPRESSION)	Forces an argument to be evaluated in scalar context. This function is useful if the expression may be a list and you don't want the expression to be evaluated in list context.
seek(FILEHANDLE, POSITION, RELATIVE)	Sets the position of the file pointer for the file. The value of the third argument specifies the relative position of the file pointer: 0 means it's relative to the beginning of the file; 1 means it's relative to the current position; and 2 means it's relative to the end of the file. The function returns true if successful.
seekdir(DIRHANDLE, POSITION)	Sets the position for the readdir function.
select(FILEHANDLE)	Selects the file handle to be used by the print and write functions if no file handle is specified in them. The function returns the file handle that was in use before the function was called. If you do not give an argument to the function, it simply returns the file handle being used.
select(READBITS, WRITEBITS, EXCEPTIONALBITS, TIMEOUT)	Tells you which of your file descriptors are ready to do reads or writes, or are reporting an exceptional condition.
semctl(ID, SEMNUM, COMMAND, ARGUMENT)	Controls semaphore messages on systems that support semaphores.
semget(KEY, NSEMS, FLAGS)	Returns a semaphore ID on systems that support semaphores.
semop(KEY, OPSTRING)	Performs semaphore operations on systems that support semaphores.
send(SOCKET, MESSAGE, FLAGSJO)	Sends a message to the socket.
setgrent	Causes the next call to getg r e n t to start reading from the top of the /etc/groups file.
sethostent	Causes the next call to get hos tent to start reading from the top of the /etc/hosts file.
setnetent	Causes the next call to getnetent to start reading from the top of the /etc/networks file.

setpgrp(PROCESSID, PROCESSGROUP)	Sets the process group for the process specified in the first argument.
setpriority(PROCESSKIND, WHO, PRIORITY)	Sets the priority for a process, a process group, or a user.
setprotoent	Causes the next call to getprotoent to start reading from the top of the /etc/protocols file.
setpwent	Causes the next call to getpwent to start reading from the top of the /etc/passwd file.
setservent	Causes the next call to get servent to start reading from the top of the /etc/services file.
setsockopt(SOCKET, LEVEL, OPTIONNAME, OPTIONVAL)	Sets an option on the socket.
shift(LISTNAME)	Shortens the list by removing the first item, and returns that item. If the list is empty, it returns the undefined value. Note: The argument must be the name of a list, not an actual list.
shmctl(ID, COMMAND, ARGUMENT)	Controls shared memory on systems that support shared memory.
shmget(KEY, SIZE, FLAGS)	Returns the shared memory ID on systems that support shared memory.
shmread(ID, VARIABLE, POSITION, SIZE)	Reads from shared memory on systems that support shared memory.
shmwrite(ID, STRING, POSITION, SIZE)	Writes to shared memory on systems that support shared memory.
shutdown (SOCKET, HOW)	Closes the socket.
sin(NUMBER)	Returns the sine of the number, expressed in radians.
sleep(NUMBER)	Causes the program to sleep for a specified number of seconds. If no argument is given, the program sleeps until it receives a S I G A L R M signal from some other program. The function returns the actual number of seconds slept.

socket(SOCKET, DOMAIN, TYPE, PROTOCOL)	Opens a socket and returns true if successful.
socketpair (SOCKET1, SOCKET2, DOMAIN, TYPE, PROTOCOL)	Creates a pair of sockets and returns true if successful.
sort { BLOCK } LIST	Returns a list of the items in the list argument sorted using the comparison specified in the block. The block compares two variables, \$a and \$b, and the comparison determines how the sorting is done.
sort(LIST)	Returns a list of the items in the argument sorted in ascending string order.
sort SUBROUTINE-NAME LIST	Returns a list of the items in the list argument sorted using the comparison specified in the subroutine. The subroutine compares two variables, \$a and \$b, and the comparison determines how the sorting is done.
splice(LISTNAME, REMOVEOFFSET, REMOVENUMBER, ADDLIST)	Returns a list that is the named list with the specified number of items removed from the specified offset with the list in the fourth argument added at the offset. Note. The first argument must be the name of a list, not an actual list.
split(/PATTERN/, STRING, LIMIT)	Returns a list that consists of a string split at each point where the pattern matches a substring of the full string. If a limit is specified, only that limited number of items is returned, with the last item being the remainder of the string after the initial segments are split out if no limit is specified, any number of items can be returned. In scalar context, the function returns only the number of items that would have been returned in list context. If the pattern contains parentheses, the delimiter matched in each set of parentheses is returned interspersed with the other list items.
sprintf(FORMAT, LIST)	Formats a list using the format string and returns the resulting list.
sqrt(NUMBER)	Returns the square root of the argument.
srand(NUMBER)	Sets the seed value for the rand function to the number specified. If you do not specify an argument, Perl uses the time function as the

argument.

stat(FILE)

Returns information about a file. The argument can be either a file handle or a string with the file's name. If the file is a symbolic link, the information is about the link itself, not the linked-to file. The following items in the list are returned: the device number of the file system, inode number, file permissions (number), number of hard links to the file, user number of file's owner, group number of the file's owner, device identifier for special files, size of file, last access time, last modify time, last inode change time, preferred block size, and number of blocks allocated.

study(SCALAR)

Carefully examines a string in order to speed up later searches on that string.

sub NAME { BLOCK }

Defines a subroutine. You also include a prototype for calling the subroutine between the name and the block. If you do not specify a block, Perl notes that the name exists when it interprets your program.

substr(STRING,
OFFSET, LENGTH)

Returns a string that is a substring of the first argument. The substring starts at the specified offset and is of the specified length, if possible. If the offset is negative, the offset is from the end of the string instead of from the beginning. If you do not specify a length, everything from the offset to the end of the string is returned. This function can also be used on the left side of an assignment, in which case the string must be a string name, and the contents of the string will change.

symlink(OLDFILE, NEWFILE)

Creates a new file that has a symbolic link to the old file. Returns true if successful.

syscall(CALLNAME, LIST)

Executes the specified system call, using the list as arguments.

sysopen(FILEHANDLE,
FILENAME, MODE)

Opens the file using system-level calls.

sysread(FILEHANDLE,
VARIABLE, LENGTH, OFFSET)

Reads from the file using system-level calls.

<code>system(COMMAND)</code>	Runs a command and waits for the command to finish. The function returns the exit status of the program multiplied by 256.
<code>syswrite(FILEHANDLE, VARIABLE, LENGTH, OFFSET)</code>	Writes to the file using system-level calls.
<code>tell(FILEHANDLE)</code>	Returns the file position keeper of the specified file.
<code>telldir(DIRHANDLE)</code>	Returns the directory position keeper of the specified file.
<code>tie(VARIABLE, NewPACKAGENAME, LIST)</code>	Associates a variable with a particular package so that calls to the package are reflected in the variable.
<code>tied(VARIABLE)</code>	Returns a reference to the object that is tied to the variable.
<code>time</code>	Returns the current time, which is the number of nonleap seconds from the time specified as the "beginning of time" for the operating system.
<code>times</code>	Returns a list of the amount of CPU seconds used. The items in the list include the number of seconds for user instructions for this process, system instructions for this process, user instructions for child processes, and system instructions for child processes.
<code>truncate(FILEHANDLE, LENGTH)</code>	Shortens the file to the specified length.
<code>uc(STRING)</code>	Returns the string in all uppercase characters.
<code>ucfirst(STRING)</code>	Returns the string with the first character changed to uppercase.
<code>umask(MODE)</code>	Sets the UNIX <code>umask</code> value for the current process.
<code>undef(VARIABLE)</code>	Undefines the value of the variable or sets the value of the variable to the undefined value.
<code>unless(TEST) { BLOCK1 }</code>	Executes a block if the test is false. The first block may be followed by <code>else</code> <code>BLOCK2</code> . In this case, if the test is true, the second block is executed. The first block may also be followed by one or more instances of <code>elsif</code> <code>BLOCKn</code> , which acts the same as <code>else if</code> .

<code>unlink(FILENAMES)</code>	Deletes the files in the list and returns the number of files deleted.
<code>unpack(TEMPLATE, STRING)</code>	Returns a list of items that is the string examined using the template (the first argument to the function).
<code>unshift(LISTNAME, LIST)</code>	Adds the items of the list in the second argument to the beginning of the list named in the first argument.
<code>untie(VARIABLE)</code>	Unbinds a variable from a package.
<code>until(TEST) { BLOCK }</code>	If the test returns false, Perl executes the block and redoes the test in order to determine whether to execute the block again.
<code>use MODULE</code>	Imports semantics from a module into the current package. You can also specify a list of arguments for the module. Perl's pragmas are also implemented with this function.
<code>utime(ACCESSTIME, MODTIME, FILENAMES)</code>	Changes the access time and modification time on the named files. The function returns the number of files successfully changed.
<code>values(ARRAY)</code>	Returns a list of the values in the associative array.
<code>vec(STRING, OFFSET, NUMBITS)</code>	Returns the value of the element of the string that is the number of bits wide, starting at the specified offset.
<code>wait</code>	Waits for a child process to terminate.
<code>waitpid(PID, FLAGS)</code>	Waits for the specified child process to terminate and returns true when the process is dead.
<code>wantarray</code>	Returns true if the current subroutine is looking for a list value or false if it is looking for a scalar value.
<code>warn(LIST)</code>	Sends a list to the standard error output.
<code>while(TEST) { BLOCK }</code>	If the test returns true, Perl executes the block and redoes the test to determine whether or not to execute the block again.
<code>write(FILEHANDLE)</code>	Writes text formatted with the format function to the specified file.

Perl's Special Variables

In the following section, Perl's special variables are described. Don't be surprised if the descriptions are somewhat puzzling. Only a few of the special variables are useful for beginning and intermediate Perl programmers.

General-purpose variables

<i>Variable</i>	<i>Description</i>
<code>\$_</code>	The default input to many functions and operations. Most useful with <code>while(<FILEHANDLE>)</code> function.
<code>\$/</code>	The input record separator. The default depends on the operating system.
<code>\$[</code>	Index of the first element in a list. The index is normally 0 (and you really shouldn't change it), but some people insist that the first item of a list should be numbered as 1.
<code>\$ </code>	Forces automatic flushing to the selected file handle if this variable is set to true. If false (the default), output to the selected file handle will only be done when the operating system feels like it or when the file is closed.
<code>\$]</code>	The version of Perl.
<code>\$0</code>	The name of the file containing the Perl program being run.
<code>\$^T</code>	Time at which this program started running.
<code>\$.</code>	The input line number of the last file handle read.
<code>\$ARGV</code>	Name of the current file, when using <code>< ARGV ></code> .
<code>@ARGV</code>	Command-line arguments for the program.
<code>@INC</code>	List of directories in which Perl looks for programs named in the <code>do</code> , <code>require</code> , and <code>use</code> functions.
<code>%INC</code>	The files that have been used by <code>do</code> and <code>require</code> . Each key is the file name specified, and each value is the full path to the found file.
<code>%ENV</code>	Operating system environment variables.

Variables that relate to errors and return values

<i>Variable</i>	<i>Description</i>
-----------------	--------------------

- \$! The current system error number or string.
- \$? Status returned by the last system function, ' (backtick) command, or | (pipe). This value is actually 256 times the number returned from the process.
- \$@ Error message returned by the last eval function.

Special variables for regular expressions

<i>Variable</i>	<i>Description</i>
\$&	The text matched by the last successful pattern match in a regular expression.
\$'	The text preceding the last successful pattern match in a regular expression.
\$'	The text following the last successful pattern match in a regular expression.
\$+	The text that matched the last successful bracketed pattern in a regular expression.
\$digit	The text matched by the set of parentheses in a regular expression. The digit corresponds to the number of the matched set.

Variabiles that relate to processes

<i>Variable</i>	<i>Description</i>
\$ \$	The Perl program's process number.
\$ <	The real userid of this process.
\$ >	The effective userid of this process.
\$ (The real groupid of this process.
\$)	The effective groupid of this process.

Variabiles for formats

<i>Variable</i>	<i>Description</i>
\$ %	Current page number.
\$ -	Number of lines left on the current page.

\$=	Page length.
\$ ~	Name of the report format.
\$ ^	Name of the top-of-page format.
A L	Text that a format puts out when doing a form feed.
\$:	Text after which a string may be broken for format continuation fields.
\$ ^ A	Contents of the accumulator of format lines.

Miscellaneous variables

<i>Variable</i>	<i>Description</i>
\$,	Output field separator, which is printed between each item in the list that is output by print.
\$ \	Output record separator, which is printed at the end of the list that is output by print.
\$ *	List separator placed between each item in a list if the list is interpreted in double-quotes, such as "@a".
\$;	Separator placed between subscripts of multidimensional arrays.
\$ ^ D	Debugging flags.
\$ ^ F	Maximum system file descriptor.
\$ ^ H	Internal compiler hints.
\$ ^ I	In-place edit extension.
\$ ^ O	Name of the operating system for which Perl was compiled.
\$ ^ P	Internal flag for the debugger.
\$ ^ W	Value of the warning switch.
\$ ^ X	Name that the Perl binary was compiled as.
@F	Holds the results of the - a command-line option.
%SIG	Signal handlers.

Perl's Predefined File Handles

<i>File Handle</i>	<i>Description</i>
--------------------	--------------------

ARGV Iterates over the filenames in @ARGV.
STDERR Standard error output.
STDIN Standard file input.
STDOUT Standard file output.
DATA Anything that follows a token `_DATA_` in a program.
_ Cache for the stat and lstat functions and file test operators.