Инструменты ввода-вывода

Потоки ввода-вывода

При запуске любой программы автоматически открывается три потока: стандартный ввод (stdin), стандартный вывод (stdout) и стандартный протокол (stderr). Поток стандартного ввода в диалоговой операционной среде связывается с клавиатурной, а потоки стандартного вывода и стандартного протокола - с дисплейной частью консоли операционной системы. Со **стандартными потоками** в Perl связываются три предопределенных **файловых манипулятора**: соответственно STDIN, STDOUT и STDERR. Связывание **имени файла** с пользовательским файловым манипулятором в программе выполняется с помощью операции **open**(), которая открывает поток обмена данными с указанным файлом и помещает в свой первый аргумент готовый к использованию файловый манипулятор.

В случае возникновения **ошибки** при открытии файла программа обычно аварийно завершается с помощью функции **die**(), которая может выводить диагностическое сообщение в стандартный поток протокола.

Имя файлового манипулятора записывается без разыменовывающего префикса и по традиции выделяется заглавными буквами.
`# открыть для чтения файл по имени, взятом из $file_name`
`open(FILE_HANDLE, $file_name) or die("Ошибка открытия файла $file_name: $!\n");`
`# или аварийно завершить программу с выдачей сообщения`
	Рекомендуется при открытии файла сохранять файловый манипулятор в скалярной переменной, что позволяет локализовать файловый манипулятор для использования только в текущем блоке или подпрограмме.
	`open my $file_handle, $file_name or die "Ошибка открытия файла $file_name: $!\n";`

Основные режимы открытия потоков ввода-вывода

Обозначение	Режим открытия	Пример
<	Чтение (существующего файла с начала), по умолчанию	`open($fh, '<','/temp/buffer.txt');`
>	Перезапись (с начала файла)	`$path='/temp/buffer.txt';` `open($fh, '>',$path);`
>>	Дозапись (в конец файла)	`open($fh, '>>','/temp/buffer.txt');`
+<	Чтение и запись (файл должен существовать)	`open($fh, '+<','/temp/buffer.txt');`
+>	Запись и чтение (файл усекается)	`open($fh, '+>','/temp/buffer.txt');`
+>>	Дозапись и чтение	`open($fh, '+>>','/temp/buffer.txt');`

Perl предусматривает при открытии потока вместе с режимом открытия указывать кодировку читаемых и записываемых данных, что позволяет автоматически преобразовывать информацию из одной кодировки в другую

```
open $in, "<:encoding(UTF-8)", 'utf8.txt' or die;
open $out, ">:encoding(cp1251)", 'cp1251 .txt' or die;
while(<$in>){ print $out $_; }
close $in or die;
close $out or die;
```

Операторы ввода данных

Команда **ввода** вывода выглядит как **<поток>.** Причем 'поток' это **файловый манипулятор (дескриптор)**, который создается функцией open(), или переменная, содержащая манипулятор. Файловый манипулятор по умолчанию — STDIN — стандартный ввод, т.е. если аргумент не указан, данные читаются из стандартного входного потока. В **скалярном контексте** читается одна строка вместе с символом '\n' перевода строки (для разделения строк файла используется **разделитель входных записей**, хранящийся в специальной переменной $/ ($INPUT_RECORD_SEPARATOR), по умолчанию \n). В **списковом** —весь файл читается в список, элементы которого суть строки файла. **В случае обнаружения конца файла** результат оператора не определен и воспринимается как false. Если не указана переменная результата, то по умолчанию это $_.

`$input = <>; # чтение строки в $input из STDIN`
`$line = <FILE>; # чтение строки в $line из потока FILE`
`$in = <$handle>; # чтение строки в $in из потока $handle`
	`say "What is your name? ";`
	`my $name = <STDIN>;# ENTER, который нажали после ввода имени, попал в переменную $name.`
	`chomp $name;# удаляем концевой перенос строки.`
	`say "Hello $name, how are you?";`

`open my $fh, "< $file" or die "Ошибка открытия: $!";` `while (my $line = <$fh>) { # чтение строки в переменную $line`	Операция чтения "кристалл" в списочном контексте возвращает список всех строк **с разделителями записей**. Так, например, можно считать файл в массив, попутно

```
chomp $line; # удаление разделителя строк
print length $line, " $line\n"; # обработка строки
} close $fh or die "Ошибка закрытия: $!";
print "done\n";
```

отсортировав его:

```
@lines= sort(<$fh>); # в @lines отсортированные строки из $fh
```

Вывод данных

Обратите внимание, что между файловым дескриптором и списком выводимых значений запятая не ставится.

```
print($list, $of, $output, $values); # вывод в STDOUT
print STDOUT $list, $of, $output, $values; # вывод в STDOUT
print(STDERR $list, $of, $output, $values); # вывод в STDERR
print FILE $list, $of, $output, $values; # вывод в FILE
print($file $list, $of, $output, $values); # вывод в $file
```

Для форматирования выводимой информации применяется функция printf(), которая преобразует выходные данные при помощи форматов преобразования. Например, так можно вывести отформатированное текущее время в разные выходные потоки:

```
my ($hh, $mm, $ss) = (localtime)[2, 1, 0];
# выбрать из списка нужные значения: часы, минуты, секунды
my $format = "%02d:%02d:%02d\n"; # формат вывода
printf $format, $hh, $mm, $ss; # вывод в STDOUT
printf(STDERR $format, $hh, $mm, $ss); # вывод в STDERR
printf $file $format, $hh, $mm, $ss; # вывод в $file
```

Задавая различные форматы преобразования, можно выводить данные в требуемом представлении или в виде колонок указанной ширины.

Ввод-вывод с произвольным доступом

При работе с данными фиксированной длины обычной практикой является считывание или запись данных в произвольном месте файла, например, при изменении только что считанного блока данных. Для этого нужно позиционировать позицию чтения или записи. Это делается с помощью функции **seek**(), которой передается три аргумента: файловый манипулятор, смещение в байтах и указатель позиции отсчета. Позиция отсчета задается числами: 0 – от начала файла, 1 – от текущей позиции, 2 – от конца файла.

С помощью функции **tell**(), которая возвращает смещение относительно начала файла, можно узнать текущую позицию чтения-записи и использовать ее для дальнейших перемещений по файлу.

```
seek($handle, 64, 0); # переместиться на 64 байта от начала
seek($handle, 25, 1); # сместиться на 25 байт вперед
seek($handle, -10, 2); # установиться на 10 байт до конца
seek($handle, 0, 0); # установить позицию в начало файла
#-----------------
$pos = tell($handle); # запомнить текущую позицию в $pos
seek($handle, $pos-5, 1); # сместиться на 5 байт назад
```

```
#увеличивается поле счетчика длиной 2 байта, расположенное в файле с позиции $new_pos
seek($file, $new_pos, 0); # установить позицию чтения
$pos = tell($file); # и запомнить ее в переменной
read($file, $number, 2); # прочитать 2-байтовое поле
seek($file, $pos, 0); # установить в исходную позицию
syswrite($file, ++$number, 2); # записать новое значение
```

Операции ввода-вывода с произвольным доступом часто используются для работы с базами данных, основанных на записях фиксированной длины, например, с файлами в формате DBF. Они позволяют организовать быструю выборку данных и запись измененных данных на прежнее место.