Name

gperl − execute Perl commands in *groff* documents

Synopsis

gperl [*file* . . .]

gperl −h
gperl −−help

gperl −v
gperl −−version

Description

This is a preprocessor for *groff* (1). It allows the use of *perl*(7) code in *groff* (7) files. The result of a *Perl part* can be stored in groff *strings* or *numerical registers* based on the arguments at a final line of a *Perl part*.

If no operands are given, or if *file* is "−", *gperl* reads the standard input stream. A double-dash argument ("−−") causes all subsequent arguments to be interpreted as *file* operands, even if their names start with a dash. **−h** and **−−help** display a usage message, whereas **−v** and **−−version** display version information; all exit afterward.

Perl regions

Perl parts in *groff files* are enclosed by two **.Perl** requests with different arguments, a *starting* and an *ending* command.

Starting Perl mode

The starting *Perl request* can either be without arguments, or by a request that has the term **start** as its only argument.

- **.Perl**

- **.Perl start**

Ending Perl mode without storage

A **.Perl** command line with an argument different from **start** finishes a running *Perl part*. Of course, it would be reasonable to add the argument **stop**; that's possible, but not necessary.

- **.Perl stop**

- **.Perl** *other_than_start*

The argument *other_than_start* can additionally be used as a *groff* string variable name for storage — see next section.

Ending Perl mode with storage

A useful feature of *gperl* is to store one or more results from the *Perl mode*.

The output of a *Perl part* can be got with backticks `` `...` ``.

This program collects all printing to STDOUT (normal standard output) by the Perl **print** program. This pseudo-printing output can have several lines, due to printed line breaks with **\n**. By that, the output of a Perl run should be stored into a Perl array, with a single line for each array member.

This Perl array output can be stored by *gperl* in either

groff strings

by creating a groff command **.ds**

groff register

by creating a groff command **.rn**

The storage modes can be determined by arguments of a final stopping **.Perl** command. Each argument **.ds** changes the mode into *groff string* and **.nr** changes the mode into *groff register* for all following output parts.

By default, all output is saved as strings, so **.ds** is not really needed before the first **.nr** command. That suits to *groff* (7), because every output can be saved as *groff* string, but the registers can be very restrictive.

In *string mode*, *gperl* generates a *groff string* storage line
```
.ds var_name content
```
In *register mode* the following groff command is generated
```
.nr var_name content
```

We present argument collections in the following. You can add as first argument for all **stop**. We omit this additional element.

.Perl .ds *var_name*
> This will store 1 output line into the groff string named *var_name* by the automatically created command
> ```
> .ds var_name output
> ```

.Perl *var_name*
> If *var_name* is different from **start** this is equivalent to the former command, because the string mode is string with **.ds** command. default.

.Perl *var_name1 var_name2*
> This will store 2 output lines into groff string names *var_name1* and *var_name2*, because the default mode **.ds** is active, such that no **.ds** argument is needed. Of course, this is equivalent to
> ```
> .Perl .ds var_name1 var_name2
> ```
> and
> ```
> .Perl .ds var_name1 .ds var_name2
> ```

.Perl .nr *var_name1 varname2*
> stores both variables as register variables. *gperl* generates
> ```
> .nr var_name1 output_line1
> .nr var_name2 output_line2
> ```

.Perl .nr *var_name1* **.ds** *var_name2*
> stores the 1st argument as *register* and the second as *string* by
> ```
> .nr var_name1 output_line1
> .ds var_name2 output_line2
> ```

Example
> A possible *Perl part* in a *roff file* could look like that:
> ```
> before
> .Perl start
> my $result = 'some data';
> print $result;
> .Perl stop .ds string_var
> after
> ```

This stores the result **"some data"** into the *roff string* called **string_var**, such that the following line is printed:
```
.ds string_var some data
```
by *gperl* as food for the coming *groff* run.

A *Perl part* with several outputs is:
```
.Perl start
print "first\n";
print "second line\n";
print "3\n";
.Perl var1 var2 .nr var3
```
This stores 3 printed lines into 3 *groff* strings. **var1**,**var2**,**var3**. So the following *groff* command lines are created:
```
.ds var1 first
```

```
.ds var2 second line
.nr var3 3
```

Authors

gperl was written by Bernd Warken ⟨groff−bernd.warken−72@web.de⟩.

See also

Man pages related to *groff* are *groff* (1), *groff* (7), and *grog*(1).

Documents related to *Perl* are *perl*(1), *perl*(7).