NAME

Net::protoent - by-name interface to Perl's built-in getproto*() functions

SYNOPSIS

```
use Net::protoent;
$p = getprotobyname(shift || 'tcp') || die "no proto";
printf "proto for %s is %d, aliases are %s\n",
$p->name, $p->proto, "@{$p->aliases}";

use Net::protoent qw(:FIELDS);
getprotobyname(shift || 'tcp') || die "no proto";
print "proto for $p_name is $p_proto, aliases are @p_aliases\n";
```

DESCRIPTION

This module's default exports override the core *getprotoent()*, *getprotobyname()*, and *getnetbyport()* functions, replacing them with versions that return "Net::protoent" objects. They take default second arguments of "tcp". This object has methods that return the similarly named structure field name from the C's protoent structure from *netdb.h*; namely name, aliases, and proto. The aliases method returns an array reference, the rest scalars.

You may also import all the structure fields directly into your namespace as regular variables using the :FIELDS import tag. (Note that this still overrides your core functions.) Access these fields as variables named with a preceding `p_`. Thus, `$proto_obj->name()` corresponds to `$p_name` if you import the fields. Array references are available as regular array variables, so for example `@{ $proto_obj->aliases() }` would be simply `@p_aliases`.

The *getproto()* function is a simple front-end that forwards a numeric argument to *getprotobyport()*, and the rest to *getprotobyname()*.

To access this functionality without the core overrides, pass the `use` an empty import list, and then access function functions with their full qualified names. On the other hand, the built-ins are still available via the `CORE::` pseudo-package.

NOTE

While this class is currently implemented using the Class::Struct module to build a struct-like class, you shouldn't rely upon this.

AUTHOR

Tom Christiansen