NAME

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

SYNOPSIS

```
use Net::servent;
$s = getservbyname(shift || 'ftp') || die "no service";
printf "port for %s is %s, aliases are %s\n",
   $s->name, $s->port, "@{$s->aliases}";

use Net::servent qw(:FIELDS);
getservbyname(shift || 'ftp') || die "no service";
print "port for $s_name is $s_port, aliases are @s_aliases\n";
```

DESCRIPTION

This module's default exports override the core getservent(), getservbyname(), and getnetbyport() functions, replacing them with versions that return "Net::servent" objects. They take default second arguments of "tcp". This object has methods that return the similarly named structure field name from the C's servent structure from *netdb.h*; namely name, aliases, port, 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 `s_`. Thus, `$serv_obj->name()` corresponds to $s_name if you import the fields. Array references are available as regular array variables, so for example `@{ $serv_obj->aliases()}` would be simply @s_aliases.

The getserv() function is a simple front-end that forwards a numeric argument to getservbyport(), and the rest to getservbyname().

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.

EXAMPLES

```
use Net::servent qw(:FIELDS);

while (@ARGV) {
    my ($service, $proto) = ((split m!/!, shift), 'tcp');
    my $valet = getserv($service, $proto);
    unless ($valet) {
        warn "$0: No service: $service/$proto\n"
        next;
    }
    printf "service $service/$proto is port %d\n", $valet->port;
    print "alias are @s_aliases\n" if @s_aliases;
}
```

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