

## NAME

Net::hostent - by-name interface to Perl's built-in gethost\*() functions

## SYNOPSIS

```
use Net::hostent;
```

## DESCRIPTION

This module's default exports override the core `gethostbyname()` and `gethostbyaddr()` functions, replacing them with versions that return "Net::hostent" objects. This object has methods that return the similarly named structure field name from the C's hostent structure from *netdb.h*; namely `name`, `aliases`, `addrtype`, `length`, and `addr_list`. The `aliases` and `addr_list` methods return array reference, the rest scalars. The `addr` method is equivalent to the zeroth element in the `addr_list` array reference.

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 `h_`. Thus, `$host_obj->name()` corresponds to `$h_name` if you import the fields. Array references are available as regular array variables, so for example `@{ $host_obj->aliases() }` would be simply `@h_aliases`.

The `gethost()` function is a simple front-end that forwards a numeric argument to `gethostbyaddr()` by way of `Socket::inet_pton`, and the rest to `gethostbyname()`.

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::hostent;
use Socket;

@ARGV = ('netscape.com') unless @ARGV;

for $host ( @ARGV ) {

    unless ($h = gethost($host)) {
warn "$0: no such host: $host\n";
next;
    }

    printf "\n%s is %s%s\n",
        $host,
        lc($h->name) eq lc($host) ? "" : "*really* ",
        $h->name;

    print "\taliases are ", join(", ", @{$h->aliases}), "\n"
    if @{$h->aliases};

    if ( @{$h->addr_list} > 1 ) {
my $i;
for $addr ( @{$h->addr_list} ) {
    printf "\taddr #d is [%s]\n", $i++, inet_ntoa($addr);
}
    } else {
printf "\taddress is [%s]\n", inet_ntoa($h->addr);
}
}
}
```

```
    }

    if ($h = gethostbyaddr($h->addr)) {
if (lc($h->name) ne lc($host)) {
    printf "\tThat addr reverses to host %s!\n", $h->name;
    $host = $h->name;
    redo;
}
}
}
```

**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