NAME

User::pwent - by-name interface to Perl's built-in getpw*() functions

SYNOPSIS

```
use User::pwent;
$pw = getpwnam('daemon')       || die "No daemon user";
if ( $pw->uid == 1 && $pw->dir =~ m#^/(bin|tmp)?\z#s ) {
    print "gid 1 on root dir";
}

$real_shell = $pw->shell || '/bin/sh';

for (($fullname, $office, $workphone, $homephone) =
       split /\s*,\s*/, $pw->gecos)
{
   s/&/ucfirst(lc($pw->name))/ge;
}

use User::pwent qw(:FIELDS);
getpwnam('daemon')             || die "No daemon user";
if ( $pw_uid == 1 && $pw_dir =~ m#^/(bin|tmp)?\z#s ) {
    print "gid 1 on root dir";
}

$pw = getpw($whoever);

use User::pwent qw/:DEFAULT pw_has/;
if (pw_has(qw[gecos expire quota])) { .... }
if (pw_has("name uid gid passwd"))  { .... }
print "Your struct pwd has: ", scalar pw_has(), "\n";
```

DESCRIPTION

This module's default exports override the core getpwent(), getpwuid(), and getpwnam() functions, replacing them with versions that return `User::pwent` objects. This object has methods that return the similarly named structure field name from the C's passwd structure from *pwd.h*, stripped of their leading "pw_" parts, namely `name`, `passwd`, `uid`, `gid`, `change`, `age`, `quota`, `comment`, `class`, `gecos`, `dir`, `shell`, and `expire`. The `passwd`, `gecos`, and `shell` fields are tainted when running in taint mode.

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 `pw_` in front their method names. Thus, `$passwd_obj->shell` corresponds to $pw_shell if you import the fields.

The getpw() function is a simple front-end that forwards a numeric argument to getpwuid() and the rest to getpwnam().

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. The built-ins are always still available via the `CORE::` pseudo-package.

System Specifics

Perl believes that no machine ever has more than one of `change`, `age`, or `quota` implemented, nor more than one of either `comment` or `class`. Some machines do not support `expire`, `gecos`, or

allegedly, `passwd`. You may call these methods no matter what machine you're on, but they return `undef` if unimplemented.

You may ask whether one of these was implemented on the system Perl was built on by asking the importable `pw_has` function about them. This function returns true if all parameters are supported fields on the build platform, false if one or more were not, and raises an exception if you asked about a field that Perl never knows how to provide. Parameters may be in a space-separated string, or as separate arguments. If you pass no parameters, the function returns the list of `struct pwd` fields supported by your build platform's C library, as a list in list context, or a space-separated string in scalar context. Note that just because your C library had a field doesn't necessarily mean that it's fully implemented on that system.

Interpretation of the `gecos` field varies between systems, but traditionally holds 4 comma-separated fields containing the user's full name, office location, work phone number, and home phone number. An & in the gecos field should be replaced by the user's properly capitalized login `name`. The `shell` field, if blank, must be assumed to be */bin/sh*. Perl does not do this for you. The `passwd` is one-way hashed garble, not clear text, and may not be unhashed save by brute-force guessing. Secure systems use more a more secure hashing than DES. On systems supporting shadow password systems, Perl automatically returns the shadow password entry when called by a suitably empowered user, even if your underlying vendor-provided C library was too short-sighted to realize it should do this.

See passwd(5) and getpwent(3) for details.

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

HISTORY

March 18th, 2000

Reworked internals to support better interface to dodgey fields than normal Perl function provides. Added pw_has() field. Improved documentation.