Linux Username Conventions

Paul Gorman

October 7, 2011

Adduser

The `adduser` script is a Debian utility for adding users.

It wraps other lower-level programs like `useradd` and `pw`.

The Problem

Unless you supply a user name that matches NAME_REGEX, adduser issues a warning:

```
paulgorman@firefly:~$ sudo adduser 1@sdf

adduser: Please enter a username matching the regular expression
configured via the NAME_REGEX configuration variable.  Use the
'--force-badname' option to relax this check or reconfigure NAME_REGEX.
```

NAME_REGEX

```
paulgorman@firefly:~$ man 5 adduser.conf

   NAME_REGEX
      User and group names are checked against  this  regular
      expression. If the name doesnt match this regexp, user and
      group creation in adduser is refused unless --force-badname is
      set.  With --force-badname set, only weak checks are performed.
      The default is the most conservative ^[a-z][-a-z0-9]*$.

NOTES
   VALID NAMES
      adduser  and  addgroup  enforce  conformity  to  IEEE Std
      1003.1-2001, which allows only the following characters to
      appear in group and user names: letters, digits, underscores,
      periods, at signs (@) and dashes.  The name may no start with
      a dash. The "$" sign is allowed at the end of usernames (to
      conform to samba).
```

(However, as we shall see, the VALID NAMES entry above inaccurately describes the standard, although the actual behavior of adduser complies with it.)

Why?

POSIX Compliance (User Name)

IEEE Std 1003.1-2001 is one of the POSIX standards ("Portable Operating System Interface for Unix")

> *3.426 User Name*
>
> *A string that is used to identify a user; see also User Database. To be portable across systems conforming to IEEE Std 1003.1-2001, the value is composed of characters from the portable filename character set. The hyphen should not be used as the first character of a portable user name.*

POSIX Compliance (Portable Filename Character Set)

3.276 Portable Filename Character Set

The set of characters from which portable filenames are constructed.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 . _ -

The last three characters are the period, underscore, and hyphen characters, respectively.

POSIX Compliance

So, POSIX compliance—and compatibility with other *NIX variants—is one reason that `adduser` limits the characters in user names.

POSIX versus NAME_REGEX

But the default NAME_REGEX is even more restrictive than POSIX portable filename character set.

```
^[a-z][-a-z0-9]*$
```

Why?

POSIX versus NAME_REGEX

Here are two notable difference between the default NAME_REGEX pattern and the POSIX portable filename character set:

- File names can begin with a number, but user names may not.
- File names can contain dots (.) but user names may not.

Why user names can't begin with numbers

UID's are numbers. Each account has a numerical user ID in addition to a user name.

```
paulgorman@firefly:~$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
games:x:5:60:games:/usr/games:/bin/sh
mike:x:1001:1001:Mike M,,,:/home/mike:/bin/bash
scott:x:1002:1003::/home/scott:/bin/bash
```

The third field in `/etc/passwd` is the UID.

Why user names can't begin with numbers

Many user management tools accept either user names or UID's as arguments.

`chown` is one tool that accepts either user names or UID's.

chown

```
NAME
       chown - change file owner and group

SYNOPSIS
       chown [OPTION]... [OWNER][:[GROUP]] FILE...
       chown [OPTION]... --reference=RFILE FILE...

DESCRIPTION
       This  manual  page documents the GNU version of chown. chown
       changes the user and/or group ownership of each given file.
       If only an owner (a user name or numeric user ID) is given,
       that user is made the owner of each given file, and the files'
       group is not changed.  If the owner is followed by a colon and
       a group name (or numeric group  ID), with  no  spaces  between
       them, the group ownership of the files is changed as well.  If
       a colon but no group name follows the user name, that user is
       made the owner of the files and the group of the files is
       changed to that user's login group.  If the colon and group
       are given, but the owner is omitted, only the group of the
       files is changed; in this case, chown performs the same
       function as chgrp.
```

Why user names can't begin with numbers

This makes root the owner of example.txt:

```
chown 0 example.txt
```

Having a non-privileged user with the user *name* zero would be very dangerous.

Even user names beginning with a number introduce some ambiguity. Names like "0cooper" might break scripts that call `chmod`, for example.

Why user names can't contain dots

chown didn't always use the semi-colon as the separator between the owner name and group name.

It originally used the dot instead, and the dot is still valid chown syntax. These two commands do the same thing:

```
chown paulgorman:paulgorman example.txt
chown paulgorman.paulgorman example.txt
```

Why user names can't contain dots

If a user was named `paul.gorman`, `chown` would assume that the following command should set the owner of the file to `paul` and the group to `gorman`:

```
chown paul.gorman example.txt
```

In practice, `chown` might do the right thing, but dots in user names introduce unnecessary ambiguity. Although colons are its default separator, removing the dot notation from `chown` would compromise compatibility.

A real world example

Today poor Kiel ran into an agonizing bug. A few weeks ago, building a custom RPM of perl-5.10.0 (that is, the Perl distribution itself) wasn't a problem. The unit tests passed with nary a care.

But today it no longer worked. I'll omit details of the many false paths Kiel had to go down in trying to figure out why an obscure test in the Module::Build package was failing. Eventually I took a look and noted that he'd tried all the logical troubleshooting. Time to look at the ridiculous. What if the test was failing because the last time he built it successfully it was under the user "rpmbuild", while he was now trying with user "rpmbuild-local"?

That was exactly the problem. Module::Build's tilde directory (˜username) parser was of the (false) opinion that usernames consist only of \w, that is, alphanumerics and underscores. The reality is that pretty much anything is valid in a username, though some characters will cause trouble in various contexts (think of / : . for example).

I explained in more detail in CPAN bug #33492 which reports someone else's experience with the test failing when the username had a backslash in it, such as the Active Directory name "RIA\dillman".

Posted by Jon Jensen

http://blog.endpoint.com/2008/08/on-valid-unix-usernames-and-ones-sanity.html

Disclaimer

This presentation is not a comprehensive explanation of the subject, and should not be relied upon as a specification for real systems.