

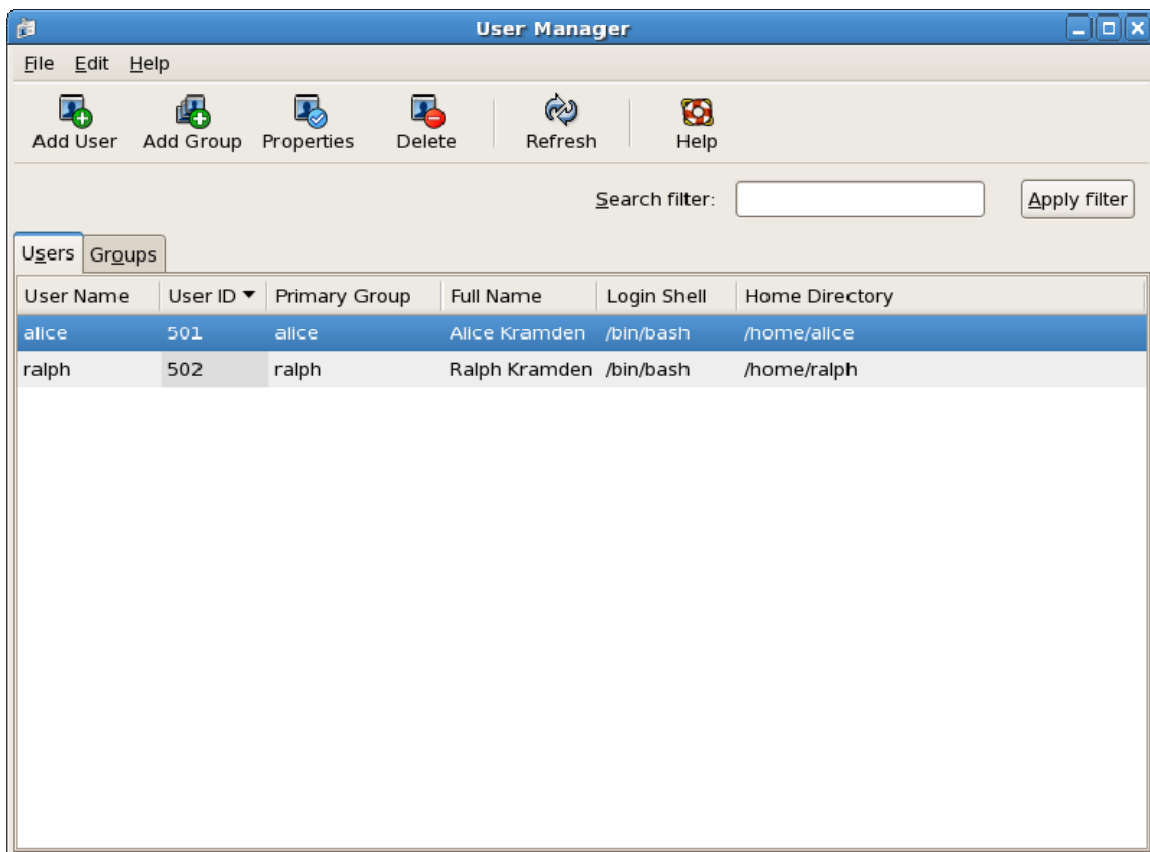
Apache UserDir Support for public_html Websites

The Apache web server has the capacity to provide user-specific directories to be accessed using the `http://example.com/~user/` syntax. A good example of a website that is hosted by alice would be <http://students.fccj.edu/~alice>.

The UserDir feature makes it possible for unprivileged users to host a website in their home directory. The Apache module that provides this feature is `mod_userdir.c`, and the Apache directive is `UserDir`. Let's explore how we can set up two unprivileged users to host websites in their home directories.

Users who host Websites

In this example, we have two users named Alice Kramden and Ralph Kramden. Alice and Ralph have basic unprivileged user accounts (as `alice` and `ralph`) in `/home` on the Linux server at `students.fccj.edu`.



There is nothing special about the `alice` or `ralph` accounts that is needed at this point.

A Basic mod_userdir.c Module

Before Alice and Ralph can actually host their websites, some reconfiguration of the Apache web server is needed. By default, most Linux distributions include the Apache webserver with the UserDir directive turned off. We have to turn UserDir on. In a typical httpd.conf configuration file, this is what the mod_userdir.c module looks like:

```
#
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.
#
# The path to the end user account 'public_html' directory must be
# accessible to the webserver userid. This usually means that ~userid
# must have permissions of 711, ~userid/public_html must have permissions
# of 755, and documents contained therein must be world-readable.
# Otherwise, the client will only receive a "403 Forbidden" message.
#
# See also: http://httpd.apache.org/docs/misc/FAQ.html#forbidden
#
<IfModule mod_userdir.c>
#
# UserDir is disabled by default since it can confirm the presence
# of a username on the system (depending on home directory
# permissions).
#
UserDir disable

#
# To enable requests to ~/user/ to serve the user's public_html
# directory, remove the "UserDir disable" line above, and uncomment
# the following line instead:
#
#UserDir public_html
</IfModule>
```

We can see by the comments that we are setting up a user's home directory for a website. When we examine the UserDir directives in mod_userdir.c, we see that UserDir is disabled. We see that the actual location of UserDir is set to public_html, which is commented out. We need to comment out the **UserDir disable** directive, and we need to uncomment the **UserDir public_html** directive.

Adding Support for public_html

The corrected mod_userdir.c module looks like this:

```
<IfModule mod_userdir.c>
#
# UserDir is disabled by default since it can confirm the presence
# of a username on the system (depending on home directory
# permissions).
#
#UserDir disable

#
# To enable requests to /~user/ to serve the user's public_html
# directory, remove the "UserDir disable" line above, and uncomment
# the following line instead:
#
UserDir public_html
</IfModule>
```

Once these changes are made, we need to restart (or reload) the Apache webserver. If we do not restart (or reload) Apache, then Apache will continue to ignore requests for UserDir websites.

Permissions for public_html

The Apache webserver does not run with root permission. Typically, the Apache webserver will have the permission of the apache user, or possibly the nobody user. We can now proceed to examine Alice and Ralph's home directories. In this example, we are going to show the permissions of /home/alice and /home/ralph.

```
[root@students home]# pwd
/home
[root@students home]# ls -l
total 8
drwxr-xr-x 4 alice alice 4096 Apr  7 13:39 alice
drwx----- 4 ralph ralph 4096 Apr  7 13:39 ralph
```

We have a difference between the permissions of the alice and ralph directories. Because of the typical 755 permissions that are set on alice, the Apache web server (running as the apache user) will be able to change directory into alice and attempt to access her public_html directory.

However, ralph is set for maximum privacy, and until ralph (or root) relaxes the permissions on the ralph directory, any attempt by the Apache webserver to access the website in ralph's public_html directory will probably return a **403 Forbidden** error message.

Alice has also set up a public_html directory in her /home/alice directory. Here is what alice's public_html directory permissions look like:

```
[root@students alice]# pwd
/home/alice
[root@students alice]# ls -l
total 4
drwxr-xr-x 2 alice alice 4096 Apr  7 13:39 public_html
```

We can see that alice's public_html is owned by alice, and that it has the typical 755 permissions that will allow other users (including apache) to read and access the contents of her public_html directory. Alice is now ready to install a website in her public_html directory. When a web surfer requests <http://students.fccj.edu/~alice> with their browser, the Apache webserver will return the website that is in /home/alice/public_html.

In future, if the administrators of the system where alice and ralph are hosted add any new users, there is no further reconfiguration of the Apache httpd.conf configuration file that is required for those new users to host their own public_html websites. Also, it is not necessary for the Apache webserver to be restarted (or reloaded) in order for Apache to serve up the websites of any new users. Apache will automatically discover new users when their public_html websites are requested by a web browser.