

Re: A handy utility (at least for me)

Re: A handy utility (at least for me)

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/hackers/2006-08/msg00422.html>

- *From:* Mike Meyer <mwm-keyword-freebsdhackers2.e313df@xxxxxxxxx>
 - *Date:* Mon, 28 Aug 2006 14:18:48 -0400
-

In <20060828164218.GA34151@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>, Rick C. Petty <rick-freebsd@xxxxxxxxxxxxxxxxxxxx> typed:

On Mon, Aug 28, 2006 at 06:18:58PM +0200, Oliver Fromme wrote:

Rick C. Petty wrote:

```
> I find that the following command works just fine for me:  
> find /usr/ports -type d -name work -prune -print -delete  
The following is probably the most efficient solution.  
It doesn't run into all subdirectories (and works with  
an arbitrary number of subdirectories).  
cd /usr/ports; echo */*/work | xargs rm -rf
```

You might as well just do:

```
rm -rf /usr/ports*/*/work
```

because using xargs doesn't gain you anything in this case. How does your example work with an arbitrary number of subdirectories?

If echo is a shell built in, then it works just fine, and the xargs insures that you don't try passing to many arguments to rm.

Your example doesn't work if the number of work directories exceeds the maximum number of arguments

That limit is in the kernel exec. If echo is a shell built in, then the kernel exec doesn't get involved until after xargs has had a chance to chop the list of arguments up. If you used "rm" instead of echo, that isn't the case.

Also I don't see how your example is any more efficient than find-- you're just making the shell do the work instead of find.

Find will check *every file* in *every directory* to see if it's named "work" or not. The shell version won't make that test on the first two

Re: A handy utility (at least for me)

Re: A handy utility (at least for me)

levels of directories; it just expands them.

In either case, it's just a sequence of opendir()/readdir(). In fact your example would start secondary processes to do the directory removal; find has this built in and thus doesn't have the overhead of process forking.

And now you get into the question of what "efficient" means. Either process is going to spend most of it's time waiting on the disk. With the find, nothing else is happening while that's going on. With multiple processes, there's a possibility that one can be working while the other is waiting on the disk, so it might take more CPU time while taking less wall clock time. Which is more efficient? [NB: This is grossly oversimplified, but you get the general idea.]

Perhaps if on an arbitrary directory tree, find may not be as efficient, but the only directories deeper than depth of two (in my example) are work directories, and they would be pruned.

You forgot the files directories that some ports have. Your version will look through those for "work", the glob won't.

<mike

--

Mike Meyer <mwm@xxxxxxxx> <http://www.mired.org/consulting.html>
Independent Network/Unix/Perforce consultant, email for more information.

freebsd-hackers@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-hackers>

To unsubscribe, send any mail to "freebsd-hackers-unsubscribe@xxxxxxxxxxx"