Re: Best way to "Clone" a Linux Hard drive?

Source: http://linux.derkeiler.com/Newsgroups/comp.os.linux.setup/2004−02/0975.html

---

From: P.T. Breuer (*ptb_at_oboe.it.uc3m.es*)
Date: 02/22/04

Date: Sun, 22 Feb 2004 23:36:35 +0100

j3r3my_t0l$m4@salmahayeksknockers.edu wrote:
> *P.T. Breuer <ptb@oboe.it.uc3m.es> wrote:*
>
> > *Simply the clone will be broken if the apache is writing logs (it's a*
> > *near certainty, not a risk, since your server is busy enough to make you*
> > *want to keep apache open).*
>
> *Logfile integrity is not necessary for my DRM requirements.*

It's not logfile integrity, but file system integrity. If the file system is left in an incoherent state (such as free space allocated but not used anywhere, or any of a dozen other obvious situations) then the file system must be repaired to a coherent state when you come to use the image for the first time. And it's very easy for that repair to take you to a state that is coherent, but not one you wanted. Such as without all of /var/log. Or with the contents of /etc/passwd (which was being read) substituted by /etc/mtab~ (which was being written).

That's why you want to put the file system into readonly mode. To do that, you have to stop all writers. That includes at LEAST all those daemons writing logfiles directly, such as apache.

You know it makes sense.

> > *There's no point in avoiding it, since killing all daemons that might*
> > *be accessing the disk is much more difficult than simply not starting*
> > *any but what you want in the first place.*
>
> *Undoubtedly, you're correct, but if I want my cake, there must be tradeoffs...*

There is no way to dynamically find out what is writing and what is not. Forget it. Not even a fuser −muv would show you everything that could be writing in a millisecond's time..

> > *Yes, you could do the same thing with a runlevel change, but you'd have*
> > *to have a much greater knowledge of init than you seem to have! You'd*
> > *have to kill all the dameons you don't want running and you don't seem*
> > *to know what those are!*

> *>*
> *> You repeatedly assume that because I do not explicity discuss something, I*
> *> am unaware of it.*

It's clear from everything you say that you are woefully ignorant of almost everything that constitutes "normal" knowledge of the operating system. It's taken this long to get you to the state of almost understanding what the two most obvious courses are, and what their tradeoffs are. It wouldn't take anyone normally competent more than a millisecond to do so. Do you really think you can fake it?

> *> You seem predisposed towards underestimating others in*
> *> order to emphasise your own knowledge.*

No.

> *> Truthfully, I find you expleasant to deal with, but are a knowledgeable*
> *> sounding board, and have offered a good idea or two (Especially passing*
> *> instructions to Lilo from the command line).*

They are not "ideas", but the most agonisingly patently obvious things. That you regard them as "ideas" illustrates the huge gulf between your estimation of your abilities and capacities and the reality of the situation. And of course, you don't think so.

Thank you for the fantastic commentary that you pay me in saying that I am at least a "knowledgable" "sounding board" for YOUR ideas. I must say that I am extremely impressed by the depth of knowledge you show, and the brilliance and profundity of your "ideas", which dazzle us all with their elegance, explicitness, and directness. Thank you.

> *> >> >> A floppy would mean there'd be a delay in the case of an unforseen hardware*
> *> >> >> reset or reboot.*
> *> >>*
> *> >> > Eh? You can only use a floppy boot when you are physically present, so*
> *> >> > you would be there to see the unforeseen.*
> *> >>*
> *> >> As mentioned, This process will be unattended. If it were unattended, and*
> *> >> I used a boot floppy*
> *>*
> *> > You couldn't. You wouldn't be there to put it in the slot.*
> *>*
> *> Goodness, thanks for pointing that out! That's why I discarded the option*
> *> in the first place...*

Nice of you to tell us, no?

> *> > I have no idea what they are, if that is what you mean! How could I?*
> *>*
> *> They have been explicitly covered in the thread.*

I see only what you write. Explicitly or otherwise.

> >> *Nonetheless, I think I have*
> >> *mini−howto, which will detail the problem, the contstraints, and the solution.*
>
> > *Oh please don't! Talk about writing a book on eating a bowl of cereal*
>
> *A bowl of serial in the dark, with both hands tied behind the back.*

What you are talking about is run−of−the−mill sysadmin−ery. It wouldn't take anyone competent more than a few seconds to implement. It doesn't require thinking about. I even wrote the script for you in the few moments I dedicated to answering you last time, to try and get it through your very stubborn and not at all perspicacious noggin that you are making a mountain out of a molehill, and for the sake of your immense ego are pretending to yourself and all the world that there is something worth thinking about here. No. There isn't. Everyone else can see it. And the puffery is not really very amusing.

> *Obviously, the regular sector cloning of a HD is simple: It it the other*
> *requirements I have that complicate the matter.*

It isn't simple, if you insist on using the hard disk at the same time. That's why one would normally not. If you insist on doing so then you have to put it in readonly mode, or your copy will be incoherent. Unfortunately, you have no good way of killing everything that holds a file descriptor open for writing. Consider for example the effect of

```
fuser −muvk /
```

That's right. Your script would likely die too. Look:

```
% fuser −muv /

                     USER PID ACCESS COMMAND
/ root 1 ....m init
                     root 404 ....m syslogd
                     root 406 ....m klogd
                     root 410 ....m watchdog
                     root 505 ....m vim
                     root 515 ....m portmap
                     root 528 ....m ypbind
                     ...
```

and are you going to guarantee that it won't die, when init will? So you say, let's take this into a runlevel where only three things run: a script, apache, and sshd.

Unfortunately for you, killing things that have been started in a runlevel requires you to back down to runlevel S first.

> When init is requested to change the runlevel, it sends
> the warning signal SIGTERM to all processes that are unde
> fined in the new runlevel. It then waits 5 seconds before
> forcibly terminating these processes via the SIGKILL sig
> nal. Note that init assumes that all these processes (and
> their descendants) remain in the same process group which
> init originally created for them. If any process changes
> its process group affiliation it will not receive these
> signals. Such processes need to be terminated separately.

and that's only an approximation. It kills things on leaving the runlevel if they have a K script in the old and no S script in the new runlevel. More or less. It's not guarranteed to kill anything more unless you do a runlevel S. And there it uses killall5:

> killall5 is the SystemV killall command. It sends a signal
> to all processes except the processes in its own session,
> so it won't kill the shell that is running the script it
> was called from. Its primary (only) use is in the rc
> scripts found in the /etc/init.d directory.

And unfortunately for you if you are not in the init session, then you will kill init if you run it. Uh uh. Mind you, it's what *I* would investigate using. But why? It's sooooooooo much easier to reboot in a well−defined mode, with whatever you want running. And it takes maybe 30s.

> > Personally, if I were doing this, I'd tar the file system across and
> > run lilo afterwards. It's perfectly automatable and requires no change
> > of run level or reboot.
>
> This is a reasonable suggestion, however, I've found that doing a system
> "hot" takes too much time, with all of the different processes competing for
> I/O.

This not so! Processes do not "compete for i/o". There will likely be only one process doing i/o at a time! That's what "system load" measures. It's the number of processes waiting on i/o.

```
 % uptime
 11:25pm up 1 day, 19:04, 5 users, load average: 0.00, 0.01, 0.00
```

There you are − zero processes "competing for i/o". And what is this mythical i/o? DIsk reads? Nobody has a disk that is being read all the time!

> By Quiescing the system, I greatly reduce the I/O, and thus the time
> required to clone.

One more interesting and vainglorious delusion, showing complete incomprehension.

A sophisticated approach to this would probably use xfs snapshots.
But I seriously doubt that you care that much. What I suspect is
that you are running an http server which makes moment to moment
changes in its configuration and content, and you want to back it up.

Only of course you're too busy to say so.

What you should do is run a failover server as a mirror, cut the
mirror, fsck the mirror, and back it up. Failing that, you can
do the xfs snapshot thing on the mirror. Failing that you can
failover the primary server to the mirror and back up the primary.

Peter