ASK KLAUS!

Klaus Knopper is the creator of Knoppix and co-founder of the LinuxTag expo. He currently works as a teacher, programmer, and consultant. If you have a configuration problem, or if you just want to learn more about how Linux works, send your questions to:

klaus@linux-magazine.com.

Two months ago, a reader wrote, "I have been told that a Linux system can "fill up" when it runs out of inode numbers because too many small files are created." The reader had two questions about inodes numbers:

- What happens to the inode number allocated to a file that has been deleted?
- Is it re-allocated to newly created files?

I gave a correct but very technical answer to this question on inode number creation. But upon further reflection, it has occurred to me that maybe what the reader really wanted to know was, "What conditions lead to a "file system full condition" (because no inode entries are left), and what can I do about it?"

It can be an amazing and annoying situation when a partition is shown as almost empty with the *df* command, yet every attempt of writing a file results in a "no space left on the device" message.

You can easily reproduce this by creating a floppy disk with a DOS or ext2 file system, then writing many small files until you get that mysterious message that says that there is no space left, even though you can actually see that your computer is lying. Consider this small Bash shell script as an example. The script quick formats a DOS floppy disk, mounts it as */media/fd0*, and then tries to fill up the available space with very small files:

```
mkdosfs /dev/fd0
mount -t vfat /dev/fd0 ⮐
/media/fd0
count=0; while true; do
 echo "What a waste of space" \
  > /media/fd0/f$((count++))⮐
  .txt || break
done
```

After running this script, you can check the allocated space on the floppy disk with:

```
df /media/fd0
```

The fact that there is still space available, yet it is impossible to use this space for adding new files, is not a "DOS-only" limitation. Had you used ext2 instead of (v)fat, you would have gotten the same result.

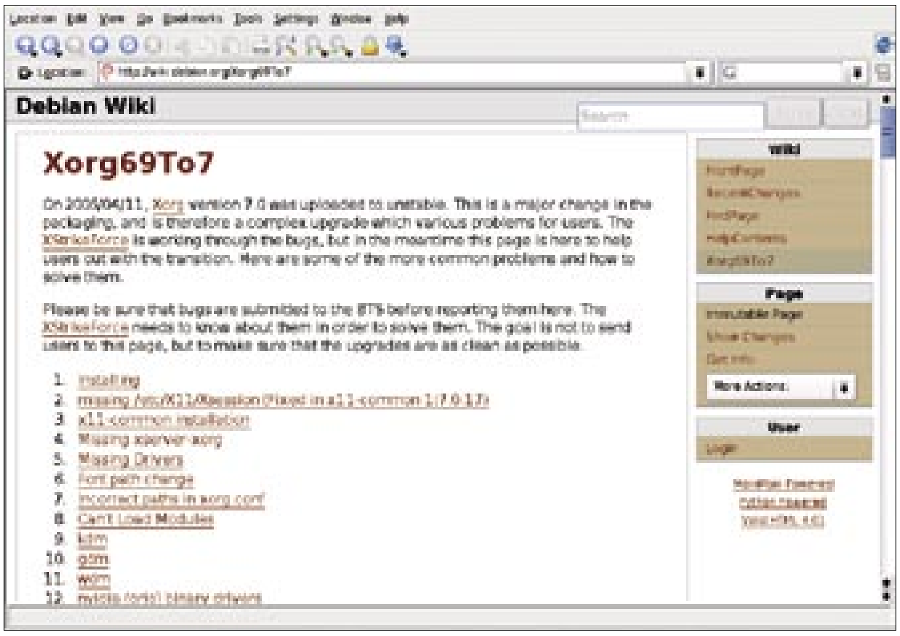
The reason is that most (older, yet most stable) file systems use a kind of static array for their "data index," the list of files and directories, and its size is determined at file system creation time (when calling mkdosfs or mke2fs). For ext2, you can easily specify the size of the file index (or, in Unix terms, the "number of inodes" that the file system may contain) using the *-N* option for mke2fs or *-r* for mkdosfs. But you can't change this fixed-size file system header later (although it is possible to resize the "data" part of an ext2 partition with *resize2fs* after changing the partition size with fdisk).

If you delete a file on a file system that has used up its available inode numbers, its empty slot will become available for a new file or directory. It depends on the file system implementation whether or not the inode number for the new file will be the same.

Most distros use a medium-sized static file index table for ext2 in order to allow a sufficient number of files/inodes without wasting too much space for empty entries. If you plan to have a very large number of small files, you should naturally increase the number of inodes, as specified by *-N* for ext2, but if you think you will store a few very large DMD image files or umcompressed video data, choosing a smaller inode number and a larger block size will waste less disk space.

btree+ file systems like reiserfs or xfs use a more or less dynamic file index that can grow and shrink according to the number of existing files, and the internal structure looks more like a tree than like a flat list of files in a continuous sequence. The advantage of this design is, of course, that there are (almost) no limitations in the number of files stored, and the way the tree is balanced makes file access very fast. On the other hand, it is more difficult to recover files if this file index tree is damaged by hardware failure, and a thorough check for data inconsistency is more complicated. (This is also related to the fact that these file systems are also using transaction logs to check, replay, or restore the last consistent state of the file system at mount time.)

NTFS and WinFS, at least according to linux-ntfs developers, seem to be using both methods: the more complicated balanced tree file index known from reiserfs plus the "old" static file index known from DOS, apparently added just for making write operations look fast in spite of the more complex file insertion methods of btree-based file systems. We are currently using the "DOS-like" file writing method of NTFS in Knoppix, which is implemented in the open source linux-ntfs driver project, so we CAN actually write on NTFS partitions without the risk of damaging its internal (and not yet fully known) structures. The kernel NTFS driver in 2.6.16/17 does not have this capability yet. I'll tell you more about that whenever the question comes up…



Xorg69To7

On 2006/04/11, Xorg version 7.0 was uploaded to unstable. This is a major change in the packaging, and is therefore a complex upgrade which various problems for users. The XStrikeForce is working through the bugs, but in the meantime this page is here to help users out with the transition. Here are some of the more common problems and how to solve them.

Please be sure that bugs are submitted to the BTS before reporting them here. The XStrikeForce needs to know about them in order to solve them. The goal is not to send users to this page, but to make sure that the upgrades are as clean as possible.

1. Installing
2. missing /etc/X11/Xsession (Fixed in x11-common 1:7.0.17)
3. x11-common installation
4. Missing xserver-xorg
5. Missing Drivers
6. Font path change
7. Incorrect paths in xorg.conf
8. Can't Load Modules
9. kdm
10. gdm
11. wdm
12. nvidia (only) binary drivers

Figure 1: Even if you aren't a Debian user, you can find good information on Linux at the Debian wiki.

config-or-fix

I have followed this Linux Magazine for several editions lately, and I was pleasantly delighted to see they had acquired you for writing some solutions to Linux problems. For what it's worth, I was even more delighted to read your fix to CD configurations and see that your recommendation was very much the same as what I had deduced and applied by my own investigation.

But can you help fix my Gentoo? I had a number of things working, and had just customized Gnome to the point where I would actually use it, when the system became inaccessible. Here is the scenario:

Boot up appears to complete to the point of login; I see a graphical login screen – either standard KDE or Gnome.

I enter the username and password, and press return. The screen changes to the selected background color for login. The system shows signs of starting a window manager (for only about 2 seconds) then aborts the login, returning to the login screen, which I suspect is an exception option in the background script.

There is only one option that works: logging in as root into a console. (I could ask how to change it so that graphical login is allowed for root, but I don't think it would be of any benefit.) This finally gives a clue as to what is the cause; an error message indicating that login just failed.

It is NOT a wrong user name or password. The regular user name with password is clearly accepted by login. It begins a *startkde* or *gnome*, but kicks out after about 2 to 3 seconds.

It looks to me like *login* is broken or corrupted. Alternatively, something is wrong at the early stages of X, KDE, or Gnome.

The system used to work perfectly well bringing up KDE or Gnome. Then some weeks ago, it stopped in the way I've described.

The only sure way I know to repair it is to reformat the partition and start again. That doesn't really fix anything; it just erases about 3 or 4 days of effort.

I'm thinking the problem may be related to X. I'm not clear on whether the login is successful. I have a feeling the login portion is broken, but perhaps the login was effective and the exception handling following some later error returned the system to the login screen. I tried reinstalling the pam_login package from a console, and it didn't help.

I think the last time I effectively completed a graphical login, it was to Gnome, which I don't normally use. You could say the system is basically intact, since you can get to a console as root, but to me, a system that fails the path to a graphical window manager is broken and needs fixing. If the system can be fixed by reinstalling, what do I need to reinstall, short of the whole system?

This problem may be easier to fix than you thought, and reinstalling Gentoo should not be necessary.

If this were a problem with *login*, or rather, the pluggable authentication methods (PAM), you would not even get to the point where the login window (kdm, gdm or xdm) disappears and starts your session.

The general procedure in a login is usually:

1. The login manager gets your password or some kind of authentication data from you, either via keyboard, smartcard, fingerprint, or whatever method you had configured in the PAM system.
2. If the authentication is verified correctly, the X server is fed with additional configuration (such as fonts, and local authentication for X window programs), and
3. Your X session is started, using your user ID.
4. If the X session ends, the X server is reset and we restart at Step 1.

From your description, I'm confident that you made it to Step 3. Xorg seems to be OK, too, unless you could identify a point where the X server crashes because of a broken font file or something similar. If you switched from Xorg 6.9 to 7.0 recently, you should read the document at *http://wiki.debian.org/Xorg69To7*.

But so far, it just looks like Step 3 terminates unexpectedly and you reach Step 4.

There are several methods for finding out why this happens. The easiest way to debug an X session is to go to the system console with Control-Alt-F1, and log in to your user account. Check the file *.xsession-errors* in your home directory to find out why the last session was terminated.

If you're looking for a more comprehensive solution:

1. Go to the system console (yes) with Control-Alt-F1, and log in as root (really).
2. Start another instance of the X server without specifying an x-cookie file:

   ```
   X :1
   ```

   The screen will turn gray, and the X server will start on a new console.
3. Go back again to your system console with Control-Alt-F1. Now, to switch to your normal user ID, type

   ```
   su - your_user_login
   ```

   as root to get a user shell. If this already fails, there is something seriously wrong with your user account. Maybe the User ID was changed for some reason, and you no longer have access to your home directory. You should get an appropriate error message in this case. If, for example, you lost your home directory to a different User ID, typing this as root will fix it:

   ```
   chown -R your_user_login ↵
   /home/your_user_login
   chmod -R u+rwX ↵
   /home/your_user_login
   ```

   (Note the case-sensitive options.) But let's assume that your home directory has the correct permissions.
4. Open a simple xterm on the X screen started in Step 2:

   ```
   xterm -display :1
   ```

   (Do this as normal user, not as root.) If you prefer gnome-terminal or konsole, you should be able to use these as well, but then, parts of Gnome and KDE could get preloaded, and the results may be different from the ones you get when you try to just log in with gdm/kdm.
5. Now, go back to the console where the new X :1 server is running. (Cycle through Control-Alt-F5…F10; it should be there somewhere.) There should now be an xterm window open where you can type commands.
6. Try to start your desired X session manually:

   ```
   startkde
   ```

   or

   ```
   gnome-session
   ```

Note that when these commands return back to the shell prompt, the X-Server would normally terminate and you would get your login window back. Not this time of course, because you started the server manually and didn't give kdm or gdm a chance to manage it.

Now you should be able to identify the point where the system usually kicks you out. Sometimes it's just a full partition (i.e., KDE or GNOME cannot create their temporary files). It may also be wrong partitions of essential system files, such as */dev/null* or an unwritable */tmp* directory.

After you have identified and fixed the error in your setup, you can just kill the running X server and go back to the login screen to retry. But wait… you should first retry your kdm login, of course.

Figure 2: Kaffeine is a good option for TV in Linux.

PCTV Card

I have a PCTV Pinnacle 50e card and would like it to work in my SUSE 10.0 machine. How do I configure it? If it's not possible to configure this PCTV card in Linux, could you recommend a PCTV card that works with Linux?

Unfortunately, I have no such card for testing, but from what I can tell, your card is supposed to work under Linux with a recent 2.6 kernel.

For watching TV and selecting channels, you can probably use Kaffeine with the DVB plugin, which is part of the current KDE 3.5.2 distribution.

You can find this information (and more about other TV cards) at *http://www.bttv-gallery.de/* ■

Send your Linux questions to klaus@linux-magazine.com.