Discussion comments

A Problem Solved

One of our number had a problem in that she expected to be asked for a password when her laptop was rebooted. Instead it logged her in immediately. If she suspended it, or locked the screen, then to restart required her password as expected. We had looked at this before and changing the obvious options iin the GUI for users failed to correct it, so something else was going on.

The first question to answer was what the Display manager was, since this is the program that controls the login screen and display. To discover that, we used a very useful little application called **`inxi`**, which is a command line tool only. Details will be given later on some of its options, but for this question we needed to run in a terminal window:

```
inxi -Sxxx
```

[**Tip**: from the keyboard the key combination Ctrl-Alt-T should bring up a terminal window. If not try Alt-T. A later section will say something about the way commands are written and the meaning of the conventions.]

The output from that command will contain a number of highlighted keywords, followed by colons, and the value of the corresponding keyword. In our case, we want the `dm` keyword, and in the actual instance this was given as `LightDM`.

We are now in a position to search on the net to see whether anyone else has had the same problem, and perhaps solved it. Using our favourite search engine, we looked for the keywords:

```
disable autologin lightdm
```

The first answer we got was to this page:

https://askubuntu.com/questions/106428/how-to-disable-automatic-login-in-lightdm

which in fact contained an answer that worked. It required the editing of the system configuration file `/etc/lightdm/lightdm.conf`, an action that needs some care.

Whenever a system file has to be changed, it is essential to make a copy, so that, in the event of an error, the situation can be reset to what it was. Personally, I have created a file in my home directory, called `Edits`, containing all the original contents of any files I have had to change. This makes editing a local and safe operation, and the only change to the system is a copy of a file to its system location.

[A word of **caution**: If your home directory is encrypted, then recovery will be much more difficult because you cannot access the encrypted form before you have logged on as that user. In such a case, it is probably better to place the files to be edited somewhere outside the home area, so it is always accessible.]

I will write the instructions here as though I was doing it on one of my machines.

Switch to the editing location:

```
cd ~/Edits
```

Copy the file to be edited:

```
cp /etc/lightdm/lightdm.conf .
```

Make another copy to act as the backup:

```
cp lightdm.conf  lightdm.conf.orig
```

Now make the edit to this text file `lightdm.conf`, probably by opening the file from your friendly file bowser, and using the text edit features. In our case it meant we had to remove the user name from a line containing `autologin-user=`.

Save the file to your local version.

Copy back into system. We are making a change to the system, so we need administrator privileges, which we can obtain by use of the `sudo` command. The first time it will ask you for your password, but, if you continue to use the same window, it will remember that for a certain amount of time (maybe an hour, it depends on how it's been set up) before asking again. To copy back, in a terminal window:

```
sudo cp lightdm.conf /etc/lightdm
```

To have it take effect required a re-boot, and it solved our problem. A password is now required after rebooting.

Options on `inxi`

This application can find and display a lot of information about the system, especially what software has been installed. It has many options that restrict the output to manageable amounts, whereas by default it produces a lot of information in several sections.

On Ubuntu-based Linux systems, if it has not been installed, it can be found in the standard repositories, and installed in the usual way.

It is a command line program and must be executed in a terminal window.

If you add the option `-x`, it will add more detail to the output, it can take up to three `x`'s, and most of the options will respond with more data.

The option `-S` provides basic information about the system. Adding `x`'s will add more to the output. Try these in turn to see the differences:

```
inxi -S
inxi -Sx
```

```
inxi -Sxx
inxi -Sxxx
```

You will see the extended output as more programs are named, and versions may also be included.

On laptops, a possibly useful one is `-B` which can tell you the state of the battery.

```
inxi -Bxxx
```

To see the basic stuff, then

```
inxi -b
```

will tell you about the system software, the machine, the CPU, the graphics cards, the network interfaces, the storage drives and general information about its current state.

To find out about the cpu (that is the engine that does all the computing) then use:

```
inxi -C
```

For full information about disk drives, try

```
sudo inxi -xxx -pl
```

and if it is UUIDs for each partition, then perhaps this is more to your liking;

```
sudo inxi -xxx -pu
```

Use of the `man` command

That was all very well for those options, but how can you find out what options are available? That is where the command `man` comes in. The name derives from the obvious shortened form of manual, and its display is of a consistent format, known as *man pages*. It is a command line command, and takes one parameter, like this:

```
man inxi
```

where you will see all the options, first listed as a terse list, and then later a proper description of each one in turn.

Options to commands almost always precede any parameters. The difference is that options control the function of the application, and what it's output looks like, whereas a parameter identifies the entity (typically a file) on which it will operate.

In Unix systems the **case of a command, an option or name of file matters**. You must pay attention to the case of what you write, whether it is upper or lower case.

If you want to know more about the `man` command, then obviously this will do the job:

```
man man
```

Get information out of Bios

Sometimes you want to know what the BIOS of a PC says. The BIOS should describe the hardware in the system and inform the software how to interface to it. The BIOS descriptors can be listed with the command line:

```
sudo dmidecode
```

The `sudo` is required, because you need administrator privileges to read the BIOS.

The output is very cryptic, and intended for engineers to use for maintenance purposes. I've put this here more for completeness than anything else.

There are options that will restrict the output to more manageable proportions. So to find out about the motherboard:

```
sudo dmidecode -t baseboard
```

The man page contains details of what is available.

You should be aware that the man page states towards the end: "More often than not, information contained in the DMI tables is inaccurate, incomplete or simply wrong." This is due to the fact that manufacturers will use a BIOS that will work for a given machine, and not necessarily tailored for the fine detail of that specific machine. Also, when a machine is repaired, the hardware may well be replaced, and then the BIOS will not have up to date information.

What the kernel knows

Because the BIOS information may be inaccurate, the Linux kernel, when it starts up, scans the local hardware to discover what is really there, as fas it can. Contents of the BIOS may be used as a last resort.

To find out what the kernel has discovered, you can look in the collection of files under the directory `/proc`. There can be found text files describing the memory, cpu, and a myriad of other things, including what every process currently running is up to.

To see the contents, use your file browser and got to the FileSystem and then the proc directory. In there, open the cpuinfo file (or whatever you want to look at).

More information on the flags in the cpuinfo file can be found at:
https://unix.stackexchange.com/questions/43539/what-do-the-flags-in-proc-cpuinfo-mean

Comments on use of commands

For those not familiar with the command line, you can open a terminal window either from the main menu, or use the key combination Ctrl-Alt-T.

Options to commands are introduced by a single – sign (minus sign or hyphen) if the option is a single character. If the option is given as a word, then two minus signs are typically used.

Every command has access to two output files. These are known as `stdout` and `stderr`, and both, by default, go to the terminal from which the command was initiated. The normal output, `stdout`, is what you usually want to see, but it is possible to redirect the output, for instance into a file which you could later edit. For instance, to put the list of your files into a text file to be edited, use this:

```
ls > list.txt
```

The ls command generates a list of files. The > sign will redirect `stdout` to the named file. You can then look at it the editor and do what you like with it.

The other standard output is `stderr`. This contains all error messages from the command and is also, by default, sent to the terminal. Again, you can redirect it somewhere else. For instance, some commands cannot read directories, only files, and so cough a message out for each directory they see. To prevent the errors swamping the real output, you can do this:

```
ls 2> /dev/null
```

The 2> magic means redirect the error messages. The special file `/dev/null` is a black hole, a write-only device. It will swallow anything and lose it permanently.

Between the > sign and the filename a blank is optional.

When writing file names, you may want to indicate your home directory. There is a special character to represent that. If the first character is a tilde, ~, then it is interpreted as the home directory. So the command

```
ls ~
```

will list the files in your home directory, no matter where you execute the command from.

When a command is executed, it has a context known as the “working directory”, and is typically shown in the prompt by the system before where you type your command. If not,then use the command to print the working directory (and perhaps unfortunately named):

```
pwd
```

which will show you where you are. Files named without a preceding / will be treated as being in the working directory.