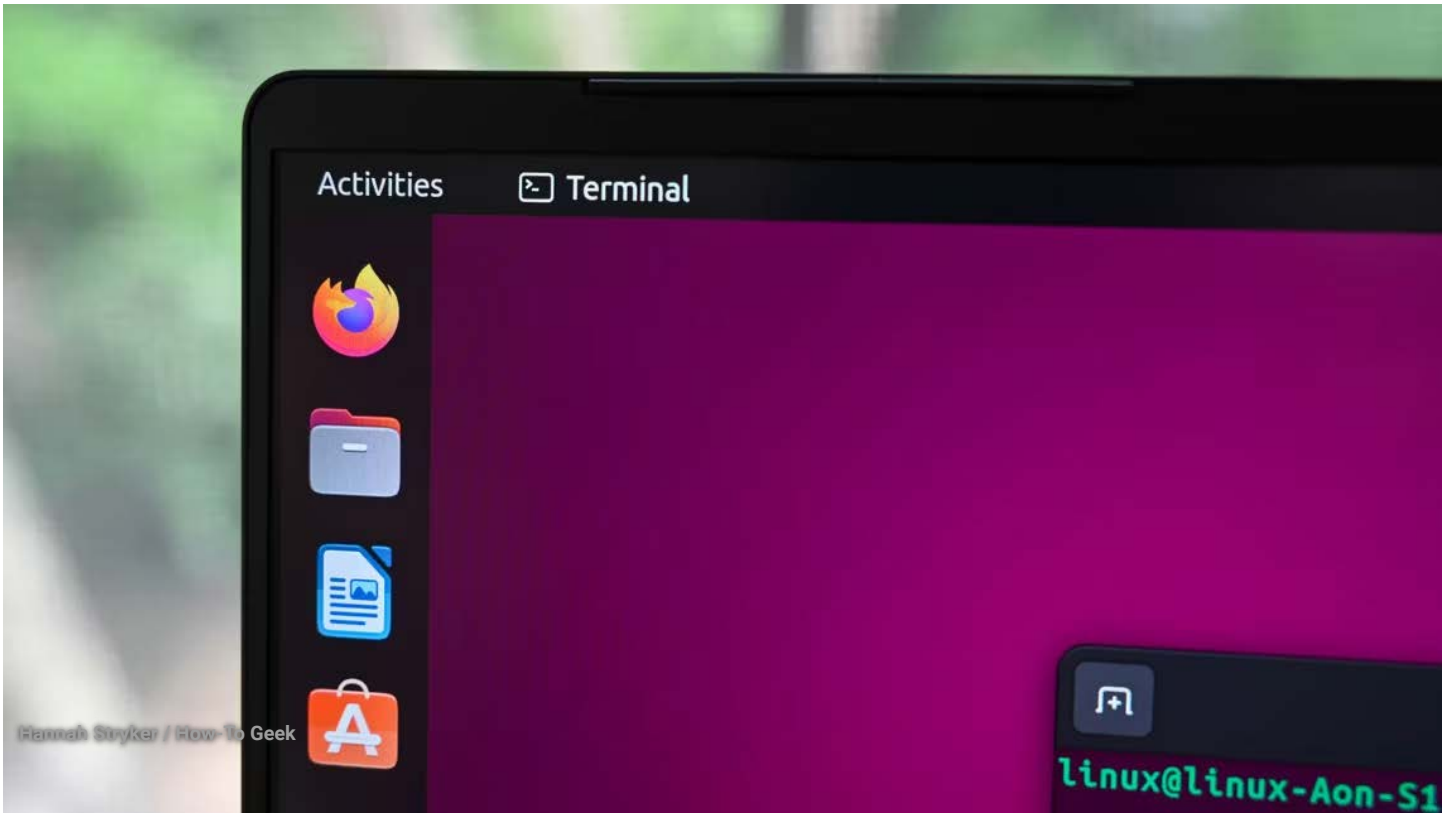
Home › Linux

How to Discover the Screen Resolution in Linux Scripts

Get a clearer picture of the display your script is working with.

BY **DAVE MCKAY** PUBLISHED SEP 22, 2023



Quick Links

KEY TAKEAWAYS

- **Use the xrandr command to query and set screen resolutions, and the output**

h awk to get the current resolution, like `xrandr | awk -F'[ +]' '/primary/{print $4}'`

- **The xdpyinfo command provides detailed information about the screen, and awk can be used to isolate the line with the current resolution, and with a little more manipulation, you can extract the X and Y resolutions as variables.**

If you need to know your current screen resolution on a Linux computer, you have several commands to choose from. Sometimes it is useful to incorporate that functionality in a Bash script.

Why Change Your Screen Resolution

Sometimes you need to know what your screen resolution is before your script launches an application or does some further processing. You may need to pass the resolution as command line parameters to the application you're about to launch, or you may have reset the resolution and you want to check that the change of resolution was successful.

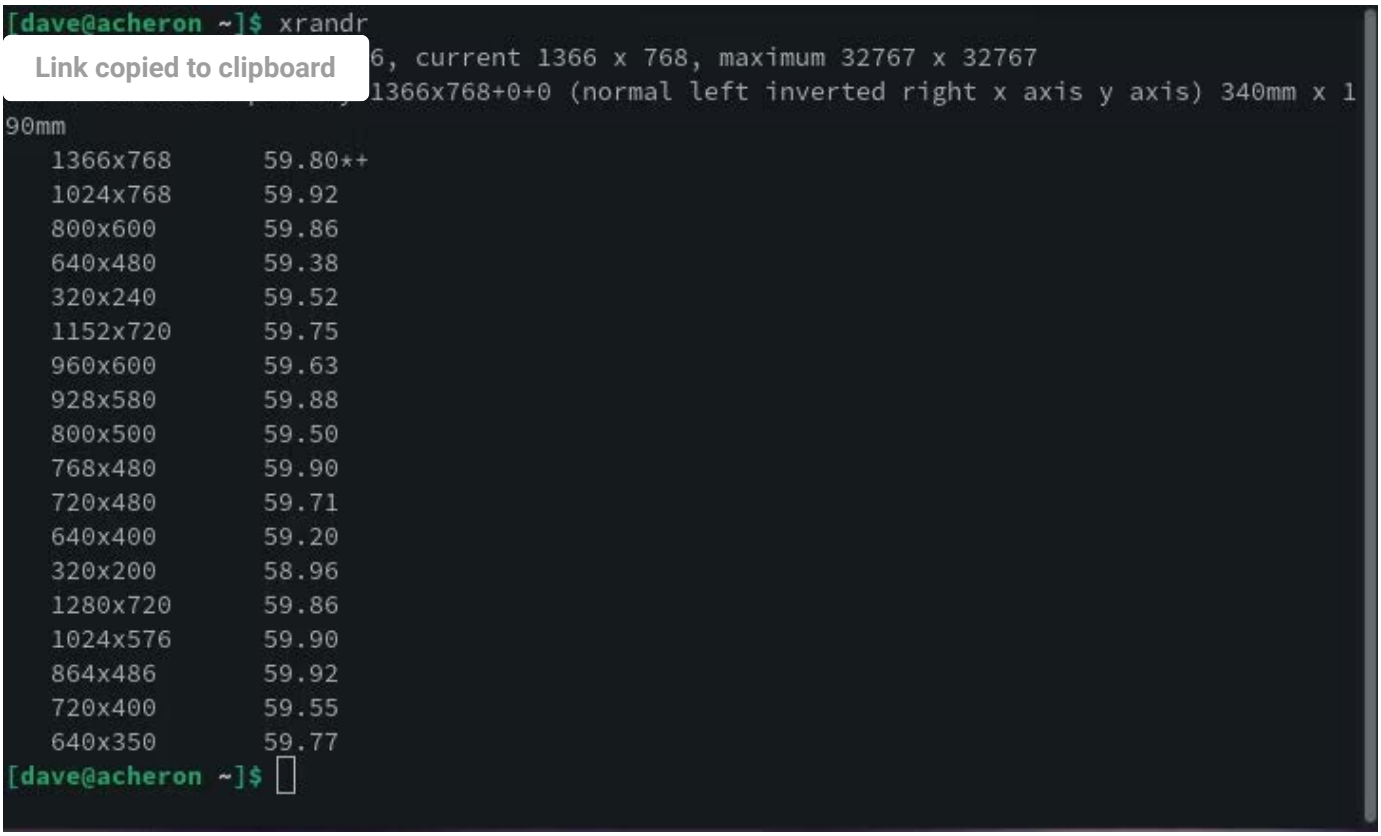
Lowering the resolution is a trick that can make playing some games on older, less powerful, hardware possible. Changing the resolution back and forth manually will soon get tiresome, so the obvious answer is to do it from within a script.

The commands we're going to look at were installed as standard on all the distributions we checked, so there shouldn't be a need to install anything. You can use these techniques straightaway, and your scripts should be portable across different distributions.

Using the xrandr Command

The xrandr command can be used to set and query the screen resolution. By default, with no parameters, it generates a list of video modes that are supported by your screen, or screens.

```
xrandr
```



The preferred video modes are followed by a plus sign “+”, and the current video mode is followed by an asterisk “*”.

This screen has only one preferred video mode, and it happens to be the current video mode. Although you can have several preferred modes, there can only be one current mode. That means if we can find the line with the asterisk in it, we can determine what the current video mode settings are.

You might see online examples where this is what they do. They use grep to find the line with the asterisk in it, and then use sed or awk to parse out the information we’re interested.

But there is a simpler way. The information in the output from xrandr contains details of the “connected primary” screen, including the current video mode resolution. We can use awk to parse that out, without using grep at all.

```
xrandr | awk -F'[ +]' '/primary/{print $4}'
```

This is how it works. We use the -F (field separator) option to tell awk that spaces and plus signs "+" are to be treated as field delimiters. We're searching for lines with the word "primary" in them. This word was picked because it appears in the line we're interested in, and doesn't appear anywhere else in the output from xrandr.

Splitting that line into fields at spaces and plus signs "+" means the current resolution is located in field four.

We use the awk print command to print the fourth field. Because we're also splitting the fields at plus signs, the "+0+0" is trimmed off the back of the resolution, becoming fields five and six. We're not interested in those.

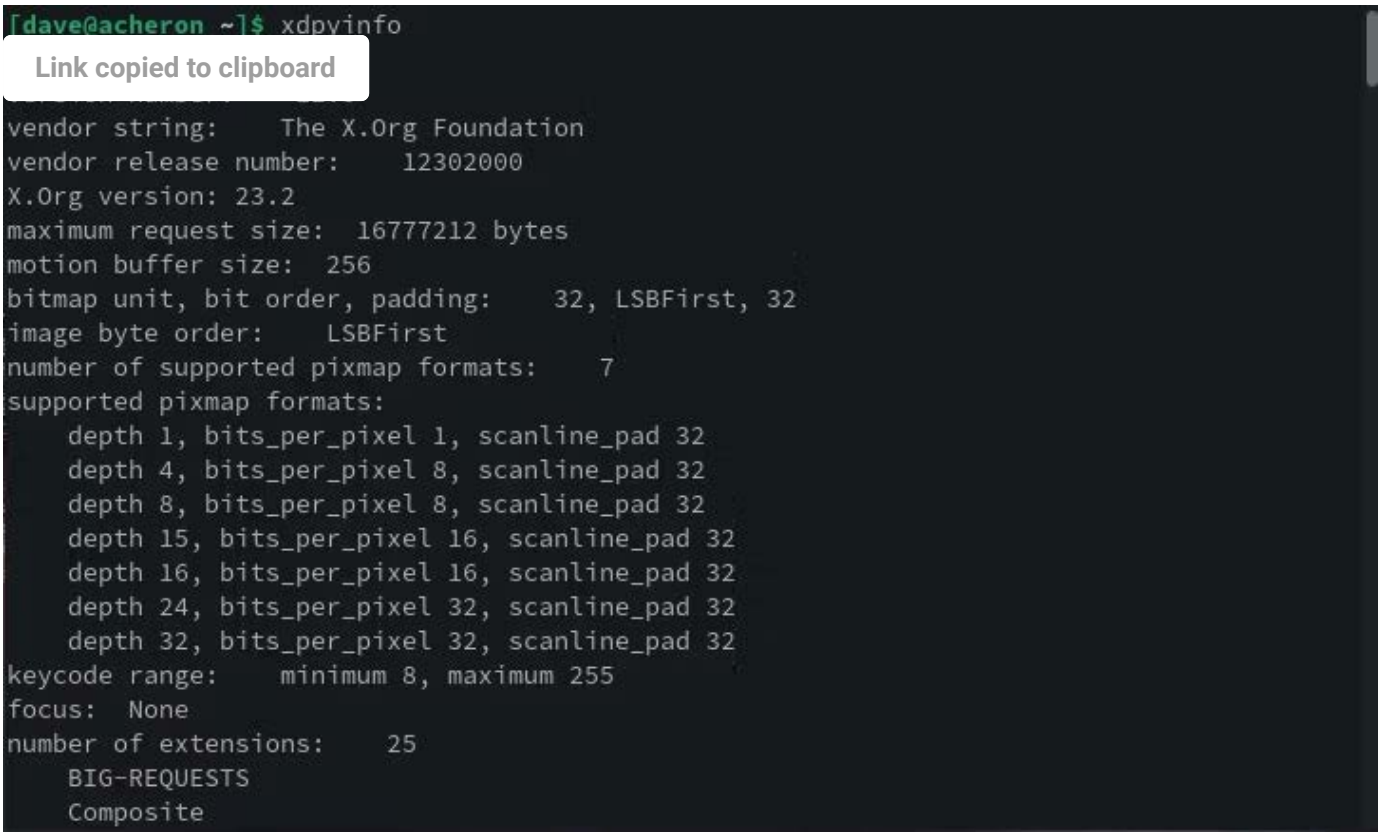
RELATED:

How To Use The Awk Command On Linux

Using the xdpyinfo Command

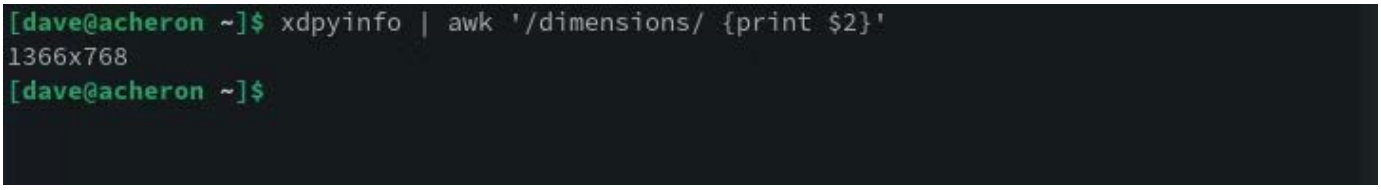
By default, the xdpyinfo command displays a lot of information about your screen and available video modes. It generates well over 1000 lines of output on our test machine.

```
xdpyinfo
```



The current mode is described in more detail than the other available modes. The line describing the resolution of the current mode includes the word “dimensions.” That word doesn’t appear anywhere else in the output, so we can use it to isolate the line of interest, using awk.

```
xdpyinfo | awk '/dimensions/ {print $2}'
```



It’s a simpler command than the one we used with xrandr, because we don’t need to specify any field delimiters. awk defaults to using whitespace as field delimiters, and that’s all that we have to contend with in the output from xdpyinfo. The resolution is the second field in the line of output, and {print $2} writes that to the terminal window for us.

Putting It Into a Script

Using either of these techniques in a script requires very little extra effort. We need to have a variable to hold our discovered resolution so that we can make a judgment in our script about

whether it ought to carry on processing or exit if the current screen resolution doesn't meet its

Copy these lines to an editor, save the file as "get_res.sh", then close your editor.

```
#!/bin/bash

current_res=$(xdpyinfo | awk '/dimensions/ {print $2}')

echo $current_res
```

We'll need to make the script executable using the chmod command.

```
chmod +x get_res.sh
```

This assigns the output of the awk command to a variable called current_res, and echoes the value of that variable to the terminal window. Let's run the script.

```
./get_res.sh
```

Encouragingly, we see the same results as we did when running the commands in the terminal window manually.

Separating the X and Y Resolutions

If you want to separate out the X and Y resolutions, we can add a couple of lines to do just that.

Link copied to clipboard

ditor, save the file as get_x_y.sh, and close your editor.

```
#!/bin/bash

current_res=$(xdpyinfo | awk '/dimensions/ {print $2}')

current_x=$(echo $current_res | awk -F'[ x]' '{print $1}')
current_y=$(echo $current_res | awk -F'[ x]' '{print $2}')

echo $current_res
echo $current_x
echo $current_y
```

We can use awk to extract the X resolution from the value stored in the current_res variable, and store the result in a variable called current_x. We repeat the process to extract the Y resolution, and assign it to a variable called current_y. Finally, we print all three variables.

Again, we need to use chmod to make it executable.

```
chmod +x get_x_y.sh
```

```
[dave@acheron ~]$ chmod +x get_x_y.sh
[dave@acheron ~]$
```

Here's the output from the script on our test machine.

```
./get_x_y.sh
```

```
[dave@acheron ~]$ ./get_x_y.sh
1366x768
1366
768
[dave@acheron ~]$
```

Having the X and Y resolutions as individual variables lets you make comparisons on each

nly the horizontal resolution that matters to you, and as long as it meets or exceeds a minimum, your script can proceed.

Keep It Sweet and Simple

Being able to isolate the vertical and horizontal resolutions lets you test either one of them, or both of them, to make sure your resolution-sensitive script can continue. It also lets you use the values as command line parameters for other applications that need to be launched with that information.

Getting hold of these values needn't be convoluted. Even a little knowledge of the standard Linux utilities such as awk will repay the time spent learning them, many times over.

Related Topics

LINUX PROGRAMMING LINUX & MACOS TERMINAL

About The Author

Dave McKay

(403 Articles Published)

Dave McKay first used computers when punched paper tape was in vogue, and he has been programming ever since. After over 30 years in the IT industry, he is now a full-time technology journalist. During his career, he has worked as a freelance…