Linux Fu: X Command

Originalartikel

Backup

<html> Text-based Linux and Unix systems are easy to manipulate. The way the Unix I/O system works you can always fake keyboard input to another program and intercept its output. The whole system is made to work that way. Graphical X11 programs are another matter, though. Is there a way to control X11 programs like you control text programs? The answer to that question depends on exactly what you want to do, but the general answer is yes. As usual for Linux and Unix, though, there are many ways to get to that answer. If you really want fine-grained control over programs, some programs offer control via a special mechanism known as D-Bus. This allows programs to expose data and methods that other programs can use. In a perfect world your target program will use D-Bus but that is now always the case. So today we’ll look more for control of arbitrary programs. There are several programs that can control X windows in some way or another. There’s a tool called xdo that you don’t hear much about. More common is xdotool and I’ll show you an example of that. Also, wmctrl can perform some similar functions. There’s also autokey which is a subset of the popular Windows program AutoHotKey. <h2>About xdotool</h2> The xdotool is probably the most useful of the commands when you need to take over GUI programs. It is sort of a Swiss Army knife of X manipulation. However, the command line syntax is a bit difficult and that’s likely because the tool can do lots of different things. Most of the time I interested in its ability to move and resize windows. But it can also send fake keyboard and mouse input, and it can bind actions to things like mouse motion and window events. Although you can make the tool read from a file, you most often see the arguments right on the command line. The idea is to find a window and then apply things to it. You can find windows by name or use other means such as letting the user click on the desired window. For example, consider this: <pre>echo Pick Window; xdotool selectwindow type „Hackaday“</pre> If you enter this at a shell prompt, you can click on a window and see the given string appear as if it were typed there by the user. The tools is also capable of sending mouse events and performing a multitude of window operations like changing window focus, changing which desktop is shown, etc. By the way, some of xdotool’s features require the XTest extension to your X server. I’ve always found this turned on, but if things aren’t working, you’d want to check your X server log to see if that extension is loaded. <h2>What About wmctrl?</h2> The wmctrl program has a lot of similar functions but mostly interacts with your window manager. The only problem is, it uses a standard interface to your window manager and not all window managers support all features. This is one of those things that makes distributing programs for Linux so exciting. No two systems are alike and some aren’t even close! The wmctrl program shines when you want to do things like switch desktops, maximize windows, and related tasks. However, it can do many of the tasks that xdotool can do, as well. <h2>Using a Big Monitor</h2> I recently switched out my three-monitor setup for a very large 4K monitor. The 43-inch behemoth has a resolution of 3840×2160. That’s great, but I did miss being able to put one program on one monitor and a second (or third) program on another monitor. The answer was to get the windows to slide into certain positions on the screen. You could use a tiled window manager, but I use KDE (which no longer has a tiling option). It will snap windows to certain positions if you drag them to just the right spot, but that’s not very fast. Also, the snap areas were not all where I wanted them. <img data-attachment-id=„273557“ data-permalink=„https://hackaday.com/2017/09/21/linux-fu-x-command/bigscreen/“ data-orig-

file=„https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png?w=800&h=450“ data-orig-size=„800,450“ data-comments-opened=„1“ data-image-meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"0"}“ data-image-title=„bigscreen“ data-image-description=„“ data-medium-file=„https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png?w=800&h=450?w=400“ data-large-file=„https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png?w=800&h=450?w=800“ class=„aligncenter size-full wp-image-273557“ src=„https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png?w=800&h=450“ alt=„“ width=„800“ height=„450“ srcset=„https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png 800w, https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png?w=250&h=141 250w, https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png?w=400&h=225 400w, https://hackadaycom.files.wordpress.com/2017/09/bigscreen.png?w=768&h=432 768w“ sizes=„(max-width: 800px) 100vw, 800px“/> My original thought was just to use xdotool and map some keys using KDE’s shortcuts. Control+Alt+1 could snap the current window to the top left of the screen and Control+Alt+0 could maximize. Control+Alt+6 would eat up the right half of the screen and Control+Alt+8 would take up the top half. My first attempt at creating the Control+Alt+1 shortcut looked like this: <pre>xdotool getwindowfocus windowmove 0 0 windowsize 1920 1080</pre> The idea is to find the current window, move it to 0,0 and then make it take up a quarter of the screen. Sure, the hardcoded numbers aren’t great, but it works for a single machine set up. You can set the size to 50% 50%, if you prefer. That makes sense for this one, but for the other macros where the position isn’t 0,0 you have to use a hardcoded number anyway. The first problem was that — in some cases — the moving wasn’t working every time. Reversing the size and the move took care of that. However, there was still a problem. If you maximize a window, the window size and position values don’t do anything. Here’s where wmctrl can help: <pre>wmctrl -r :ACTIVE: -b remove,maximized_horz,maximized_vert ; xdotool getwindowfocus windowsize 1920 1080 windowmove 0 0</pre> This removes the maximized property from the active window and then applies the xdotool command. However, if you are going to use wmctrl, you might as well say: <pre>wmctrl -r :ACTIVE: -b remove,maximized_horz,maximized_vert -e 0,0,0,1920,1080</pre> The -e option moves the window. The first zero isn’t a typo. It sets the “gravity” of the window and is usually zero. The next four numbers are the corner coordinate and the size. However, you will notice that while xdotool moves the top left corner of the window, wmctrl moves the top left corner of the interior of the window (that is, not including the window decorations). So the result is slightly different. <img data-attachment-id=„273558“ data-permalink=„https://hackaday.com/2017/09/21/linux-fu-x-command/half/“ data-orig-file=„https://hackadaycom.files.wordpress.com/2017/09/half.png?w=800&h=450“ data-orig-size=„800,450“ data-comments-opened=„1“ data-image-meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"0"}“ data-image-title=„half“ data-image-description=„“ data-medium-file=„https://hackadaycom.files.wordpress.com/2017/09/half.png?w=800&h=450?w=40

0“
data-
large-file=„https://hackadaycom.files.wordpress.com/2017/09/half.png?w=800&h=450?w=800“
class=„aligncenter size-full wp-image-273558“
src=„https://hackadaycom.files.wordpress.com/2017/09/half.png?w=800&h=450“ alt=„“
width=„800“ height=„450“ srcset=„https://hackadaycom.files.wordpress.com/2017/09/half.png
800w, https://hackadaycom.files.wordpress.com/2017/09/half.png?w=250&h=141 250w,
https://hackadaycom.files.wordpress.com/2017/09/half.png?w=400&h=225 400w,
https://hackadaycom.files.wordpress.com/2017/09/half.png?w=768&h=432 768w“ sizes=„(max-
width: 800px) 100vw, 800px“/> Of course, you could write a simple bash script to
manage all this and work out the math so if your screen size changes you don’t have to
change each macro. You could even adjust to make the windows not overlap or do other special
effects. For example: <pre class=„brush: bash; title: ; notranslate“ title=„“>#!/bin/bash

 # Change to suit or read them from xrandr
 #SCREENX=3840
#SCREENY=2160

 # If you don't have xrandr, awk, or yours doesn't put the right
 #
format out, just hardcode up top
 SCREENX=`xrandr -q | awk -F'[,]+' '/current/ { print $8
}'`&#13; SCREENY=`xrandr -q | awk -F'[,]+' '/current/ { print $10 }'`

 # Could adjust the
actual locations here if you wanted

 HALFX=$[1)]
 HALFY=$[2)]

 if [$# -ne
1]
 then
 ARG=„?“
 else
 ARG=„$1“
 fi
 case „$ARG“ in
nw)
 TOP=0
 LEFT=0
 W=$HALFX
 H=$HALFY
 ;;
 n)
TOP=0
 LEFT=0
 W=$SCREENX
 H=$HALFY
 ;;
 ne)
 TOP=0
LEFT=$HALFX
 W=$HALFX
 H=$HALFY
 ;;
 w)
 TOP=0
LEFT=0
 W=$HALFX
 H=$SCREENY
 ;;
 center)
 TOP=$[3)]
LEFT=$[4)]
 W=$HALFX
 H=$HALFY
 ;;

 e)
 TOP=0
LEFT=$HALFX
 W=$HALFX
 H=$SCREENY
 ;;

 sw)
TOP=$HALFY
 LEFT=0
 W=$HALFX
 H=$HALFY
 ;;

 s)
TOP=$HALFY
 LEFT=0
 W=$SCREENX
 H=$HALFY

 ;;

se)
 TOP=$HALFY
 LEFT=$HALFX
 W=$HALFX
 H=$HALFY
 ;;

 *)
 echo „Usage: winpos (nw, n, ne, w, center, e, sw, s, se)“
 exit 1
;;
 esac

 # do it
 # wmctrl -r :ACTIVE:-b
remove,maximized_horz,maximized_vert -e 0,$LEFT,$TOP,$W,$H
 # or here's another way
(note, this will show title bars without adjustment
 # the above method will cut them off at the
top part of screen
 wmctrl -r :ACTIVE: -b remove,maximized_horz,maximized_vert
 xdotool
getwindowfocus windowsize $W $H windowmove $LEFT $TOP

 exit 0
 </pre>
With this script, your keyboard macros can just call the script with a tag like “ne”
(Northeast) or “center” to control the window position. Any changes are easy to
manage in the script instead of spread over multiple macros. <h2>Summary</h2>
There’s a line in one of the Star Trek movies (the real ones, with William Shatner) where
Kirk tells someone that you have to learn how things work on a starship. Linux is much the same.
There’s not much you can’t do if you can only figure out how and wade through the
myriad tools that might be what you want. Sometimes it takes a combination of tools and dealing with
the infinite variety of configurations is tough, but you can usually make things happen if you try.
</html>

1)
SCREENX/2
2)
SCREENY/2
3)
$SCREENY/4
4)
$SCREENX/4