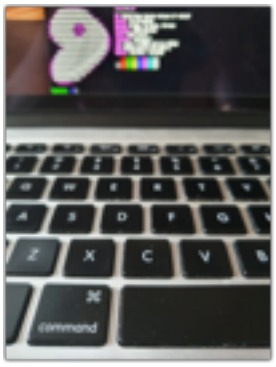
Getting MacOS style hotkeys working in GNU/Linux

Over the years I've used every operating system. Sometimes all at once. Recently I've taken to using Linux as my primary desktop operating system. Everything I need runs natively on Linux and because I have full control over my environment I never have down time due to forced updates. This is my desktop after all so I'm really switching from Windows where the Control key modifier paradigm is more or less the same as Linux. Still I didn't want to compromise on anything with this new configuration and there was one thing missing from my Macbook that I really wanted.

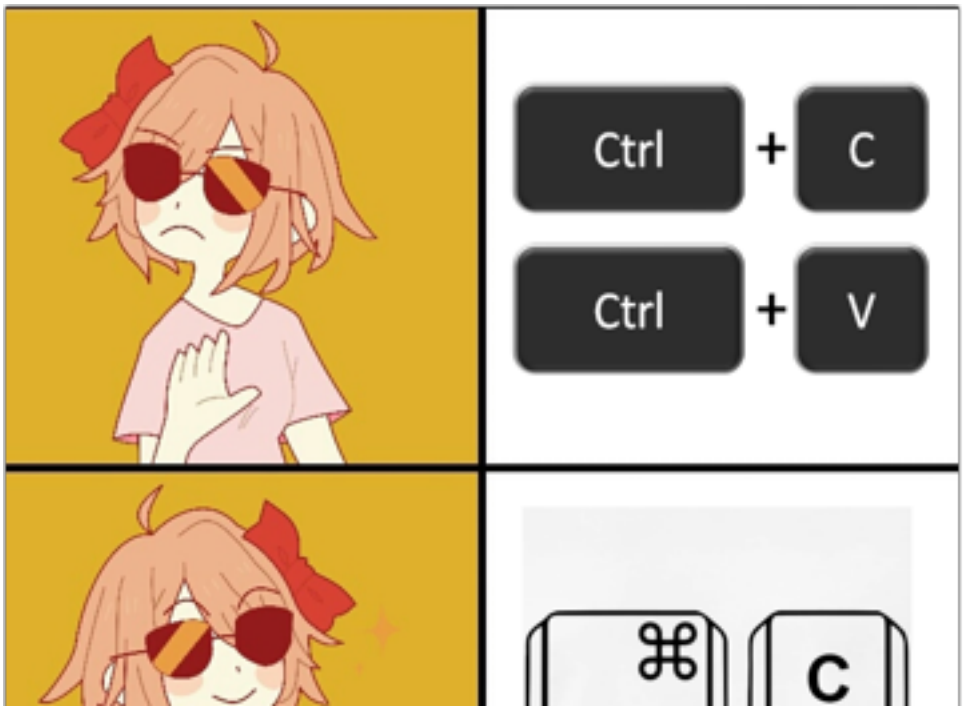
Of course what I'm talking about is the Command key or more colloquially known simply as the Apple Key and all the shortcuts and hot keys built into macOS that make it such a joy to use.



> *"Linux lets you change whatever you want."*
>
> Is a common thing that people say to bring up a positive aspect of GNU/Linux. Is this actually true though? Take for example my issue at hand: Changing copy/paste hotkeys for all programs. In truth, some tasks are so big in their scope few users would ever be capable of accomplishing them. At least not without a lot of contextual knowledge.
>
> Getting macOS style hotkeys is actually an extremely difficult task in Linux. Especially when you want to avoid faking it by binding some keys to other keys on a per application basis.

Then there's the question of why do it at all?

The truth is I just got tired of accidentally sending SIGINTs when switching to terminal and then accidentally triggering the wrong thing when switching out of terminal. I've just gotten so used to not having to context switch my brain in macOS between programs. It's one of the things that truly makes my Macbook Pro more pleasant to use.



So that's my first reason. Muscle memory.

My second reason is simply that Command/Meta is directly next to the space bar and I just like having a shorter stretching span for my hand.

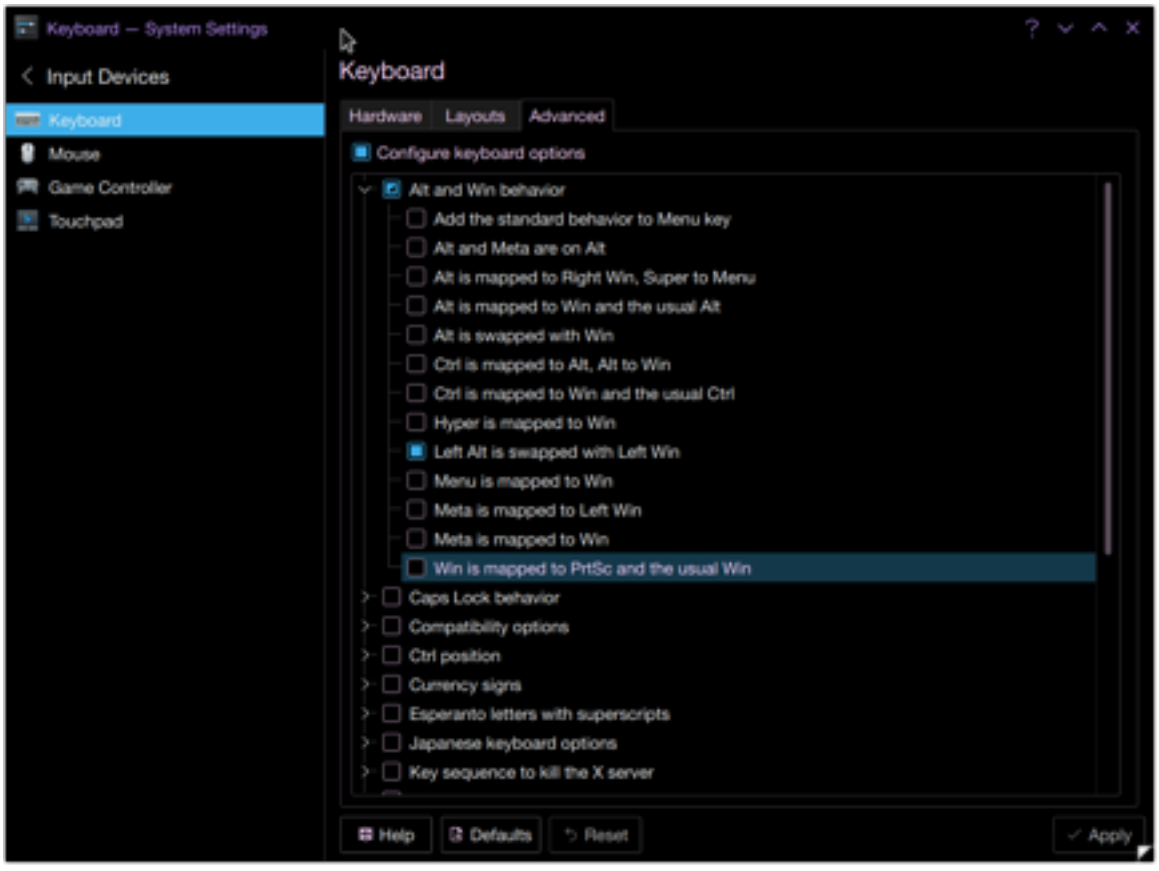
Making it work

For the most part GUI applications written for Linux tend to make a lot of assumptions about what hotkeys can and should be in a very opinionated way. Some types of programs require full customization (like code editors) while others like browsers tend to force certain hotkeys. Desktop managers do provide their own hotkey bindings. For example most KDE based applications will use your settings from KDE's "System Settings" to set hotkeys. Meanwhile in Gnome you can use dbind to set your keybinds.

Of course Cut/Copy/Paste is not enough. It would actually be quite jarring to use CTRL+T for opening a new tab immediately after using Command+V to paste something. Indeed if you go down this path you'll end up having to redefine your hotkeys for well..... everything. It is unfortunate as well that applications these days are extremely hostile to user settings.

Despite the hurdles in front of me I decided to try to make it work.

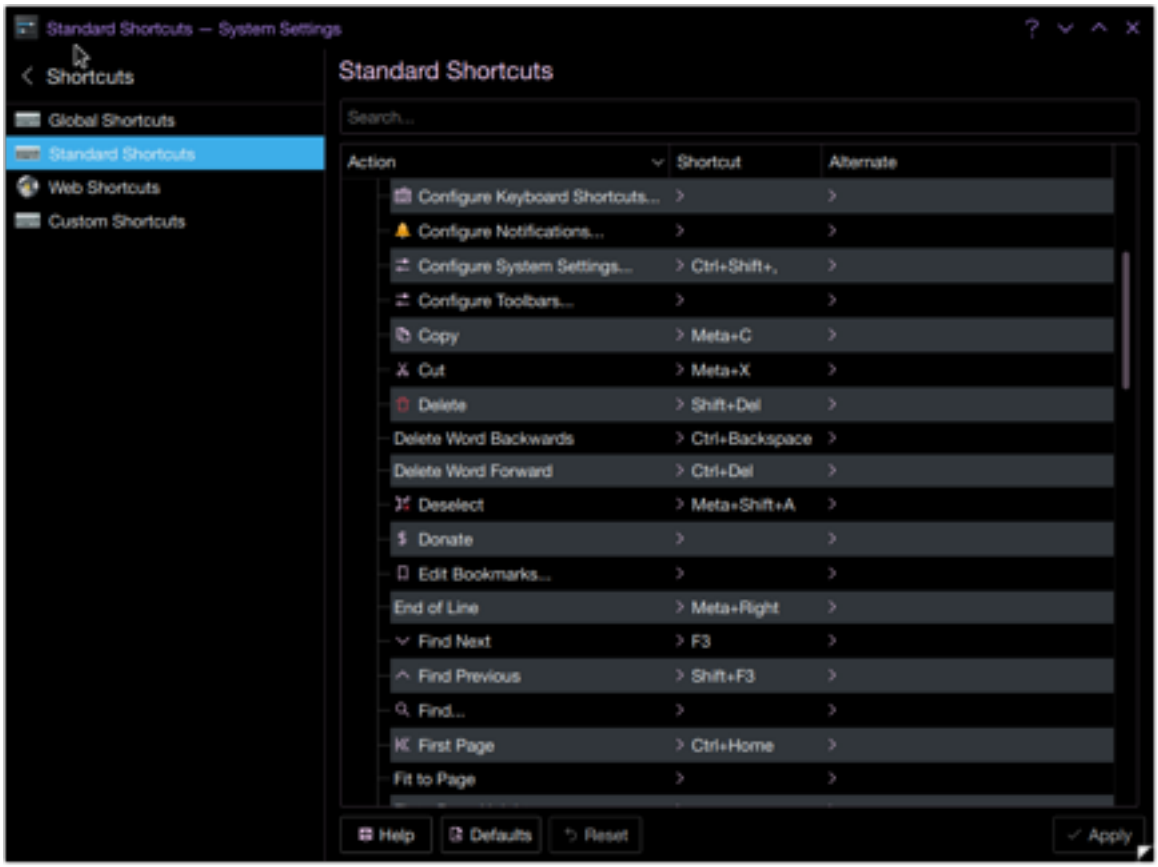
The first thing I did was swap Left Alt and Left Meta on my keyboard. This was actually pretty easy to do with keyboard mapping configs with the KDE GUI simply known as "System Settings". This was simply to make my normal windows keyboard place the buttons in the positions a real mac keyboard would have. None of this is necessary if you have a real Apple layout keyboard. This setting is only active while in my X session so any alternative TTYs would revert back to my stock layout. If I had a real Apple layout keyboard though this is not necessary at all.



This setting is saved to the text file `~/.config/kxkbrc`.

```
$ cat kxkbrc
[Layout]
DisplayNames=
LayoutList=us
LayoutLoopCount=-1
Model=pc101
Options=altwin:swap_lalt_lwin
ResetOldOptions=true
ShowFlag=false
ShowLabel=true
ShowLayoutIndicator=true
ShowSingle=false
SwitchMode=Global
Use=true
```

Now with this setting I am able to fake the same key mapping as a real macOS keyboard would have in my desktop session. So far so good. Now I manually changed all my hotkeys in my code editors, terminal applications, and even in the System Settings app I just mentioned to set global hot keys for all KDE applications. This took some time but was very straight forward for the most part.



I went above and beyond even changing my shortcuts for things like Select All.

While this worked for KDE applications I had to also make this change for GTK+ based programs as well. Luckily this would also cover Chrome, Discord, Elements, and a number of other Electron based programs that I use on a regular basis.

I was lucky that VSCode let me change everything but it was very annoying have to redo all the system based hotkeys like Set cursor to start of the line (in macOS this is Meta+Left Arrow) for each and every place I wanted to use it.

Some websites also use Meta+C for example in Github hitting Meta+C will actually take focus and bring you to the Add Comment UI when in a thread so sometimes on Github.com even with Firefox set to use Meta+C to copy something I am not actually able to use my hotkey because the website overrides it.

With GTK I ended up editing `` `~/.config/gtk-3.0/gtk.css` `` and setting these hotkeys.

```
@binding-set gtk-super-cut-copy-paste
{
        bind "<super>x" { "cut-clipboard" () };
        bind "<super>c" { "copy-clipboard" () };
        bind "<super>v" { "paste-clipboard" () };
        bind "<super>a" { "select-all" (1) };
        bind "<super>z" { "undo" () };
}

* {
        -gtk-key-bindings: gtk-super-cut-copy-paste
}
```

I might add more but for now this got me to where I wanted with most GTK+ applications. Note that if you plan to run any of programs designed to run as root that use GTK+ such as GParted you should also link or copy this config file to the root account home directory as well.

So in the end it is possible. I was able to achieve what I wanted with all but one program. But it was a huge pain. It's really difficult to figure out what settings do what in a Linux environment even when there is documentation. It's typical for documentation to exist for say GTK+ 2.0 and GTK+ 3.0 but there is no over arching best practice guide for what one should do when they want to cover all programs in an environment.