BASHing data
Data ops on the Linux command line

For a full list of *BASHing data* blog posts, see the index page. RSS

How to choose special characters, revisited

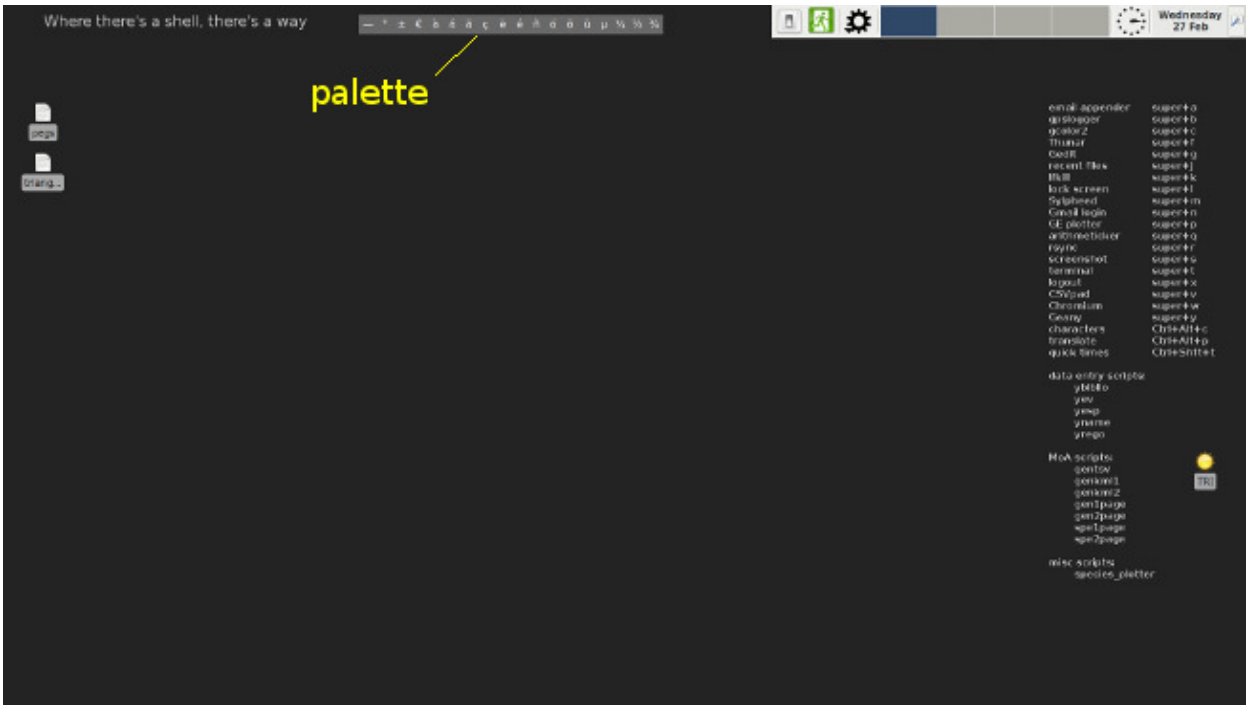
There's no euro symbol (€) on my keyboard, but I can enter that character in any document or in my terminal with *Ctrl + Shift + u +20ac*.

I can do the same with "umlaut a" (ä; *Ctrl + Shift + u +00e4*) and "cedilla c" (ç; *Ctrl + Shift + u +00e7*) and the degree symbol (°; *Ctrl + Shift + u +00b0*) and...

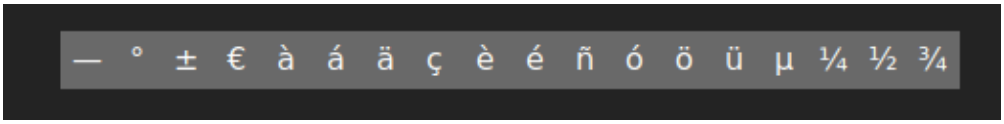
Wait! Who am I kidding? There's no way I can remember all those Unicode code points. And I don't want to waste time looking them up, or starting up a program like gucharmap, then searching for the special character I want before copying it and pasting it into the current document or terminal.

For this reason I wrote a script for quick and easy retrieval of my most-often-used special characters from a GUI. The first version was described in a 2014 blog post on Andrew Powell's *The Linux Rain* website. Here I explain the current version and its limitations.

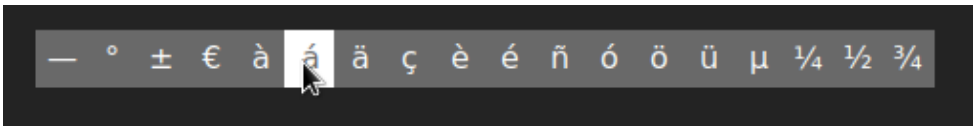
The script is launched with the keyboard shortcut *Ctrl + Alt + c*. A palette of characters appears at the top of my Xfce desktop, slightly above the panel line whre it won't be covered by any application windows:



The palette looks like this, life-size:



When I move the mouse cursor over the panel, each character is highlighted in turn:



If I left-click on a highlighted character, the character is copied to the X (primary) clipboard and can be middle-click pasted as many times as I like, in any application. The palette remains on the desktop (and can be used to copy/paste another selected character) until I right-click anywhere on the palette, at which point it disappears.

The script uses **dzen2**, which is in all the main repositories:

```
#!/bin/bash -i

printf "—\n°\n±\n€\nà\ná\nä\nç\nè\né\nñ\nó\nö\nü\nµ\n¼\n½\n¾\n" \
| dzen2 -x 550 -y 15 -w 470 -bg "#696969" -fg white -h 30 -sa c -p -l 18 -m h
```

A newline-separated list of characters is piped to **dzen2** using **printf**. The **dzen2** options are:

```
-x 550          Position the pallete 550 px from the left of the screen
-y 15           Position the pallete 15 px from the top of the screen
-w 470          Make the palette 470 px wide
-bg "#696969"         Palette background is a medium gray
-fg white         Character colour is white
-h 30           Line height (= font height + 2 px, by default)
-sa c           Slave window (sa) is centered (c)
-p          Palette will persist until closed
-l 18           Palette holds 18 items
-m h            dzen2 in horizontal (h) menu (m) mode
```

The 3 **dzen2** defaults used here are

- default font and font size (different choices could be specified)
- "Reverse highlighting" of menu choice on mouse hover
- Left-click action is to launch an application (see below)

Left-clicking on the euro character (€) in the palette launches the 1-line euro script, named "€", which sends a euro character to the X clipboard with **xclip**:

```
#!/bin/bash

printf "€" | xclip
```

There are similar short scripts for each of the other 17 special characters, all in a folder in my $PATH.

Note that the script uses the BASH *-i* flag, which puts the shell in interactive mode. Without this setting, the script won't work.

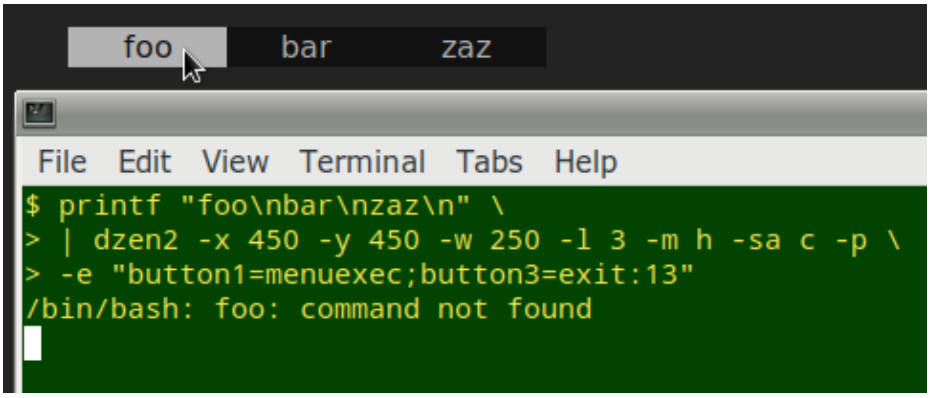
There are some limitations to this app. One is that if I want to add a new character to the palette, I have to write a new **xclip** script for it, and change the *-l 18* option in the **dzen2** command to *-l 19*, and maybe adjust the palette width. That's not a lot of work, though, and I don't change the pallete very often.

Two more important limitations have to do with how **dzen2** has been coded. The default behaviour for a horizontal menu can be spelled out with an option as:

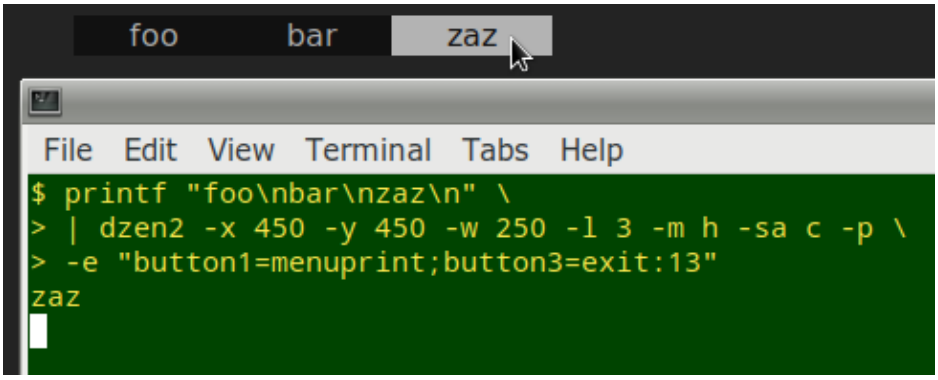
-e "button1=menuexec;button3=exit:13"

This means that clicking the left button (button 1) triggers the "menuexec" action, which launches a program in my $PATH. Because this is a shell action, the selected item must be the name or alias of a real program, which is why I have 18 little executable scripts, one for each special character. I've tried to find a simpler way to do this, so far without success. Suggestions would be welcome from any BASH wizards who read this post; here's a practice menu to play with:

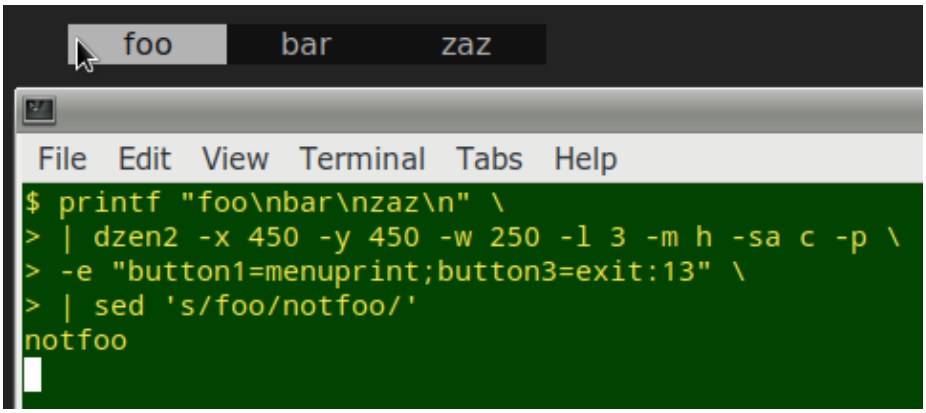
printf "foo\nbar\nzaz\n" | dzen2 -x 450 -y 450 -w 250 -l 3 -m h -sa c -p -e "button1=menuexec; button3=exit:13"



A horizontal menu can also use the "menuprint" action, which prints the selected item to stdout:



You might be thinking that's a way to send the selection directly to **xclip**, making all those 1-character scripts unnecessary. After all, once a selection goes to stdout, it can be piped to another command:



However, piping to **xclip** (or to **xsel**) fails. I'm guessing that **dzen2** owns the X primary selection in "menuprint" mode while it's waiting for further input.

Last update: 2019-03-24