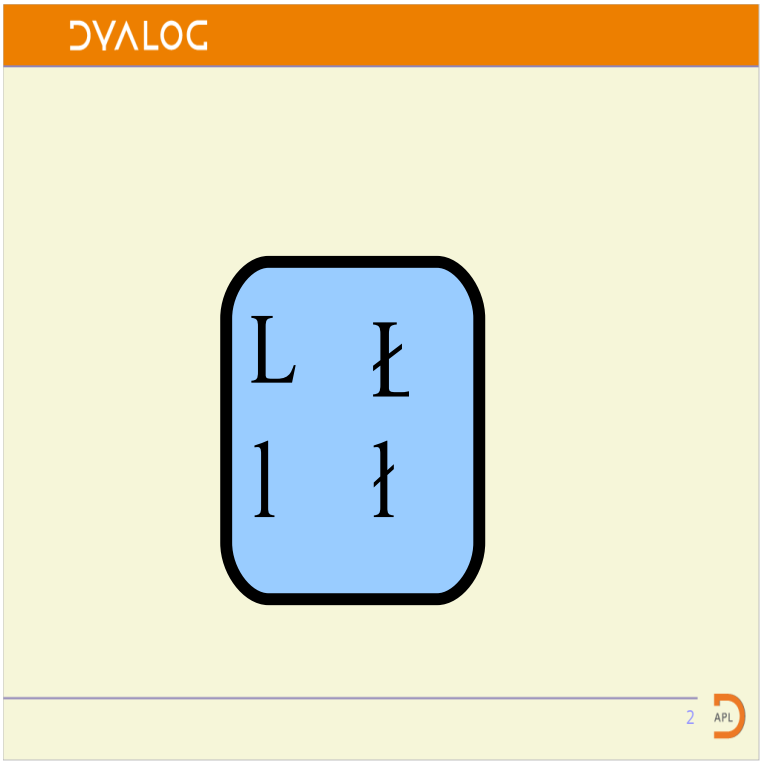
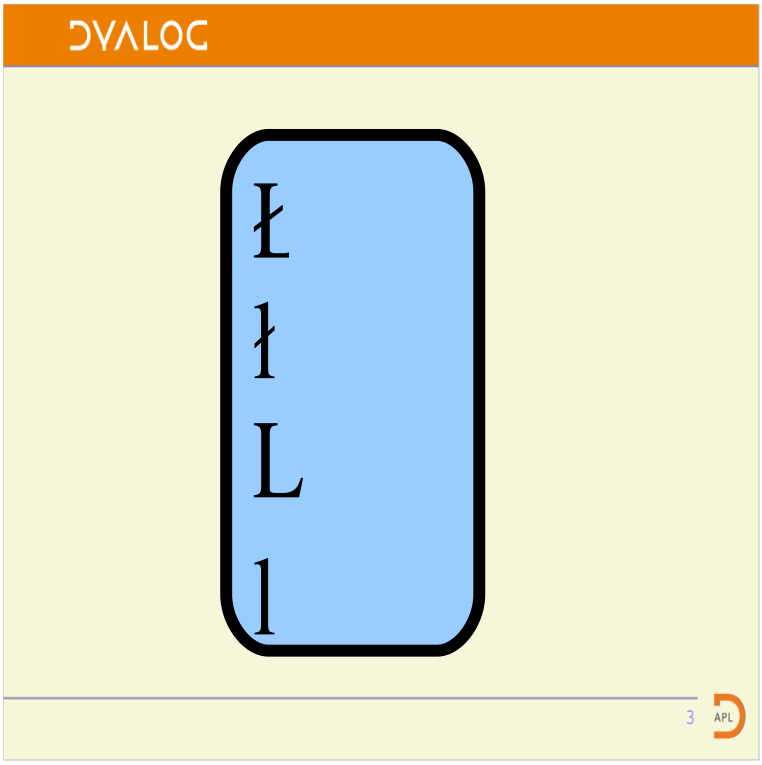
DYALOG

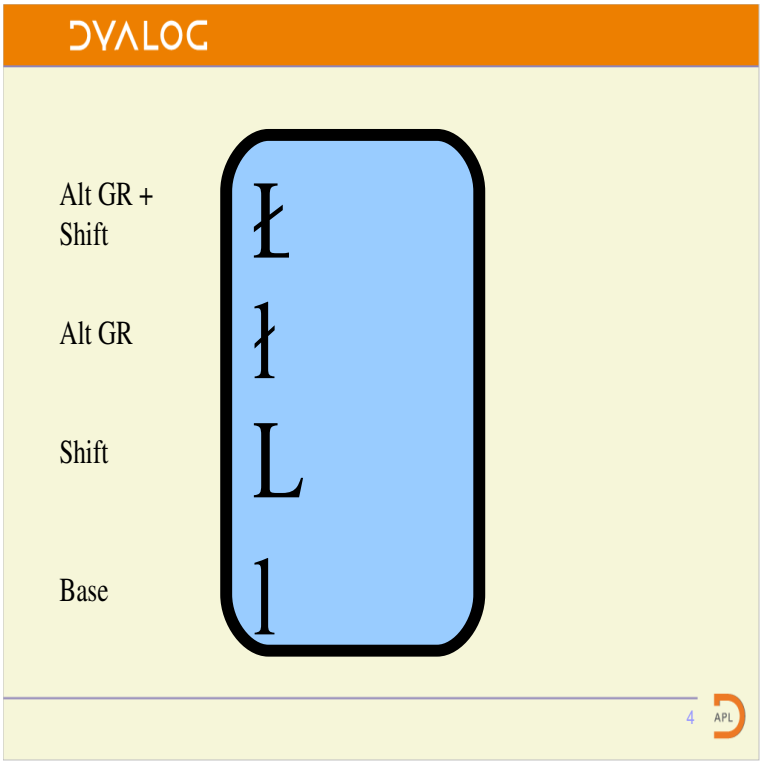
Geoff Streeter
Dyalog Ltd



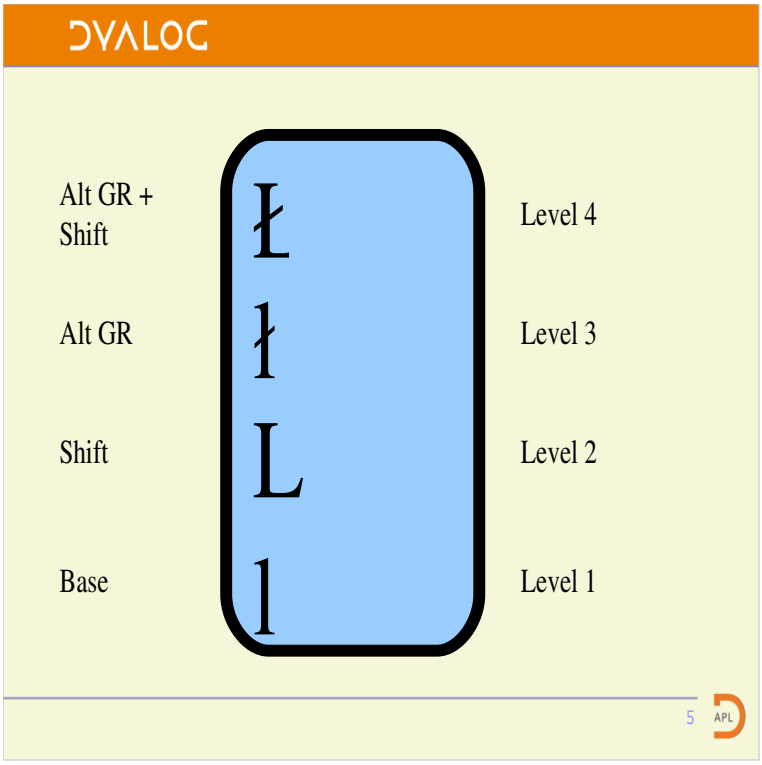
As befits a corporate minion kowtowing to the corporate management. Not only have I got the right background but I have picked a key from a Danish keyboard.



Now the fact that the key annotation was arranged as a matrix was purely an artistic convenience. So I have applied my own artistic license and re-arranged it.



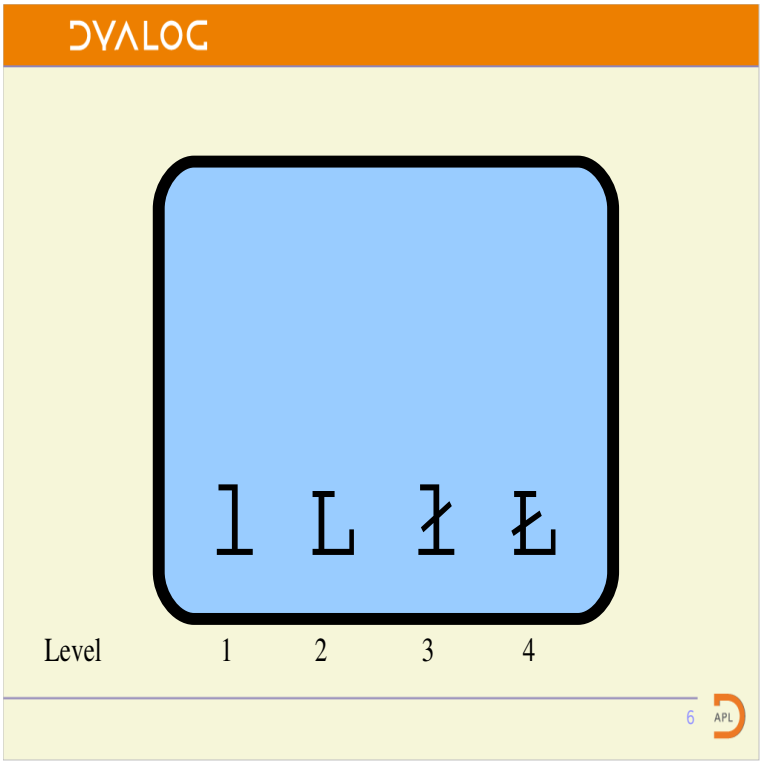
We have to have a way of choosing which character is chosen when the key is pressed.



At this point – 1996 – along comes two things. ISO9995 which is a standard for keyboards that I really ought to read, but haven't, because I baulked at the number of Swiss Francs required to get the PDF.

The other was some work by Erik Fortune of Silicon Graphics who did the X keyboard extension using ISO concepts.

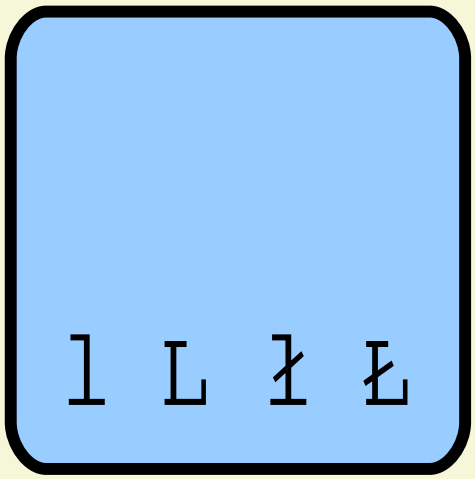
The shift states are called “levels”. They are normally drawn like a lift indicator. Or since the origin is American an elevator indicator. The same ancestry presumably also gave the origin 1 nomenclature.

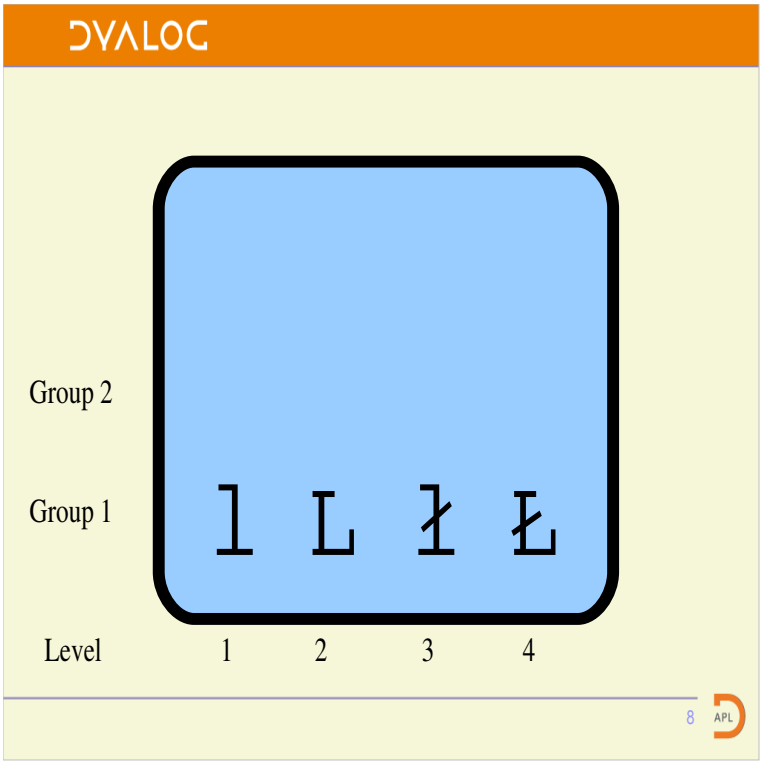


We are array people so transposing it is no problem and makes it easier to manage on a wide screen.

XKB allows 256 levels. So maybe I need an even wider screen.

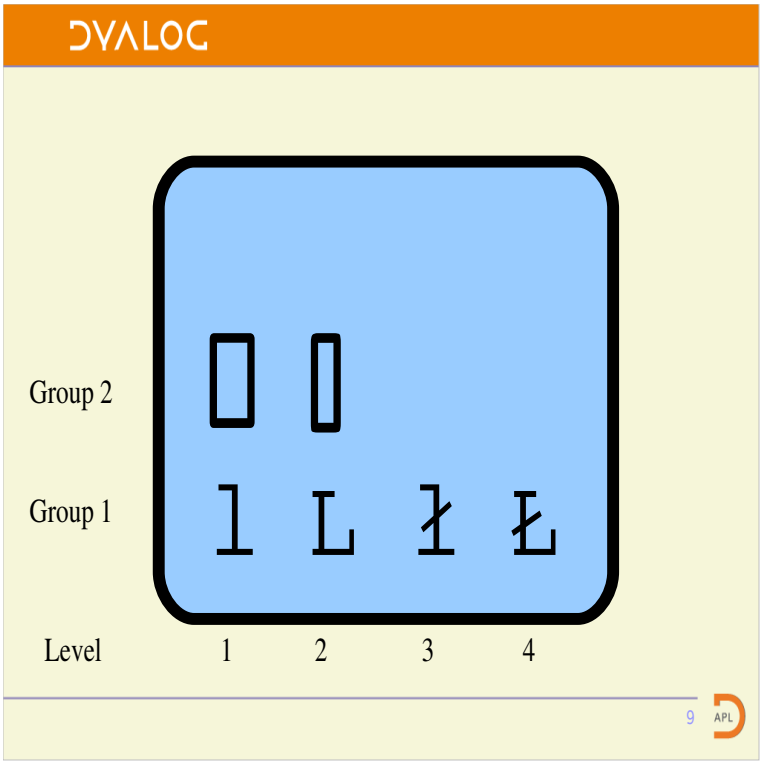
Now why have I left all that space at the top?


Group 1
ł Ł ł Ł
Level
1
2
3
4



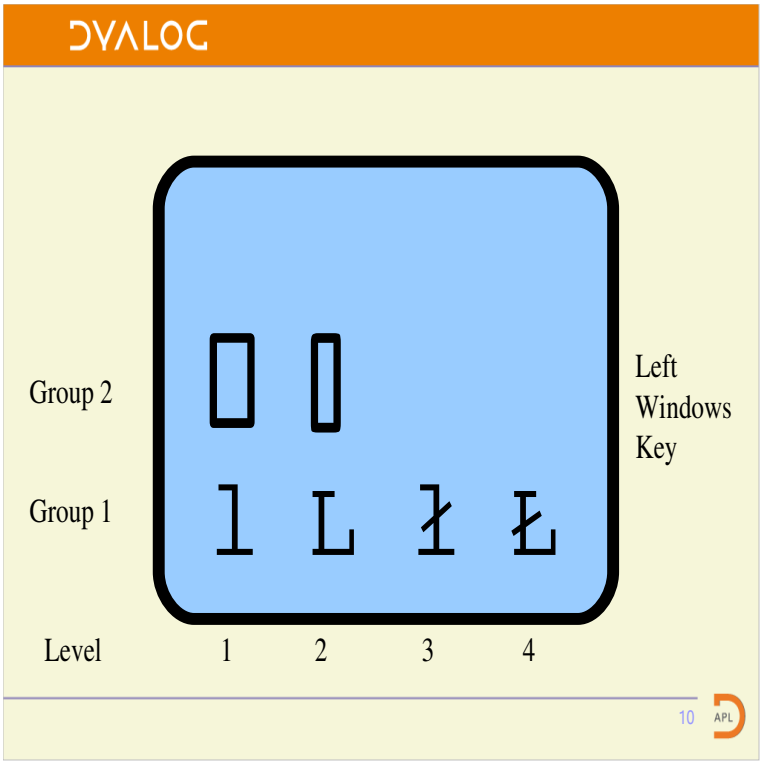
Because ISO9995 and XKB have another trick up their sleeves.

Groups.

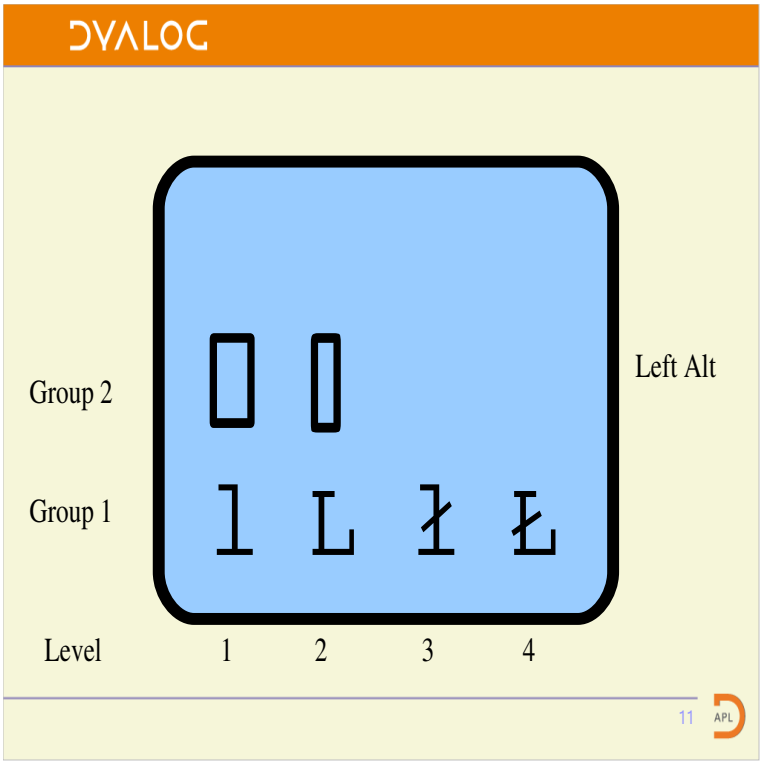


We can use another group to introduce APL's own characters.

There can be four groups. So if you are really mad you can use 256 levels and 4 groups to put 1024 characters on one key.



There are several pre-packaged strategies for changing groups. On this laptop the most convenient was the left windows key normally known in Linux circles as Super L



In the office I have an old clacky IBM keyboard with real APL keycaps and on that I choose to use the left Alt key.

DYALOG

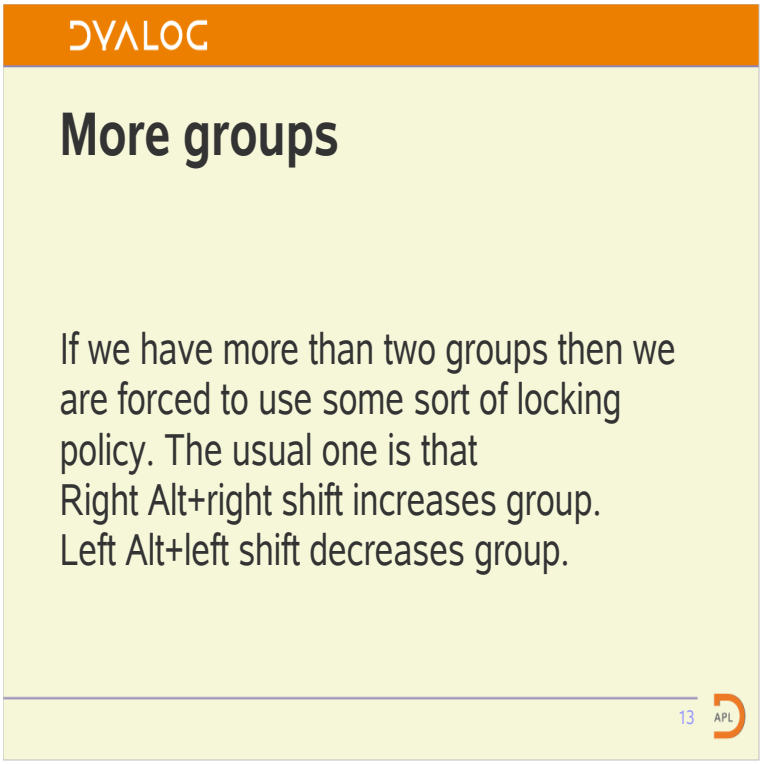
Group Shift Techniques

Locking - press and release - it stays selected. Like Caps Lock.

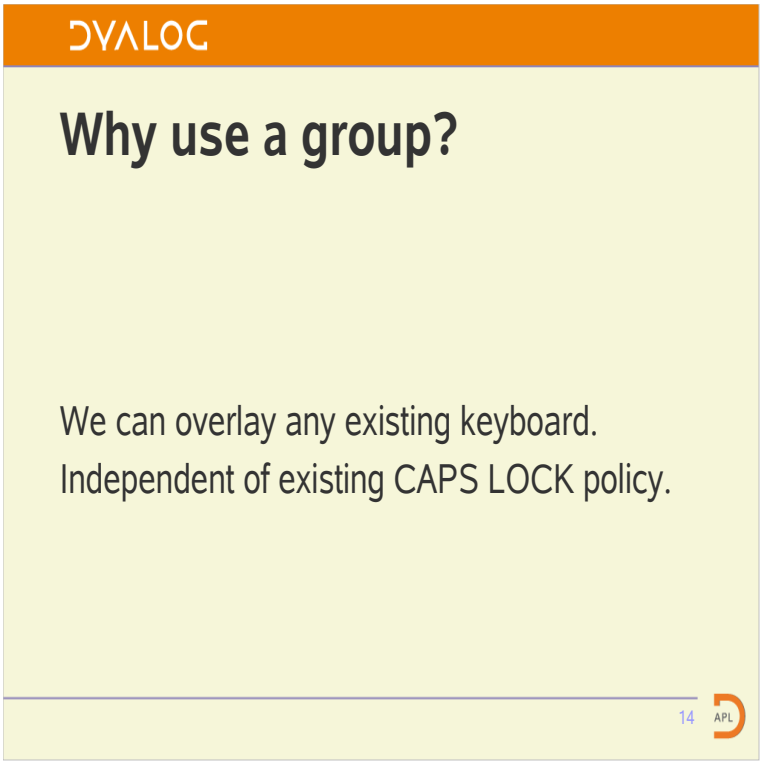
Latching – operative whilst pressed. Like shift.

- If we only have two groups then a latching policy is preferred

For most APL usage we only want one APL character at a time so we would definitely prefer to use a latching policy.



This is OK if you are going to type a word, or a sentence but is not so nice just to get one character.



In general we don't have to worry about the underlying language.

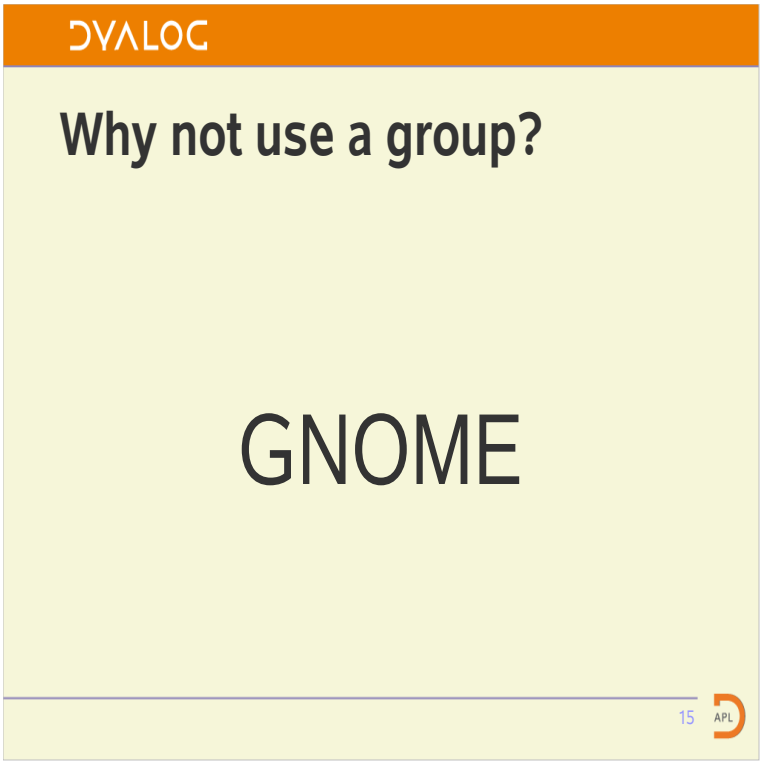
This is not quite true; the French like to move the ⍺ with the A. However they leave the ⍵ in the same place so it now sits on the Z.

What is true is that the variations are small and easy to code.

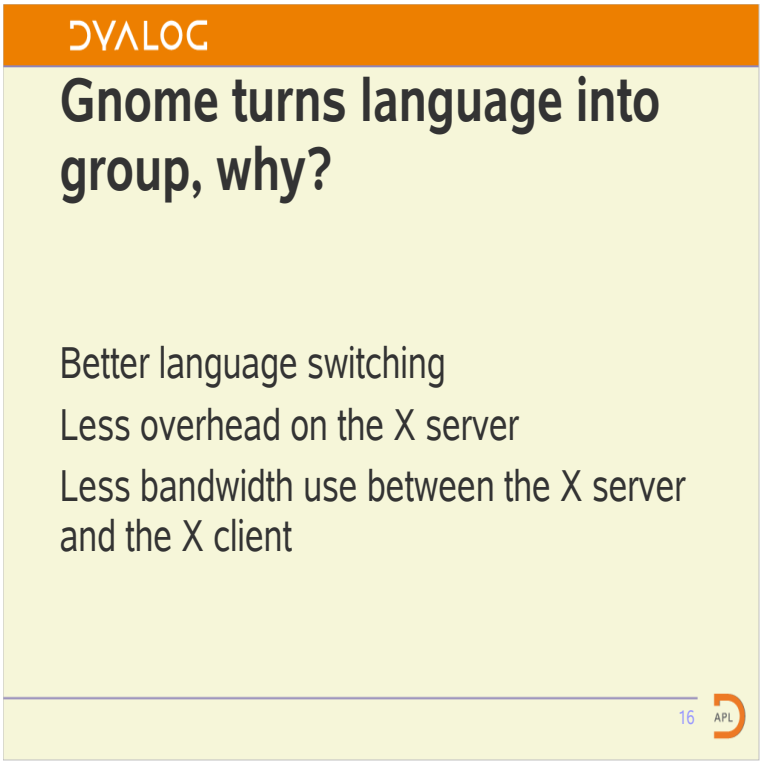
We don't want CAPS LOCK to change quad to squish.

This is actually about a concept called “type” which I haven't talked about. When we have a new group we can impose a new key type independent of the existing key type.

We can add the APL symbols to all of the layouts in use.



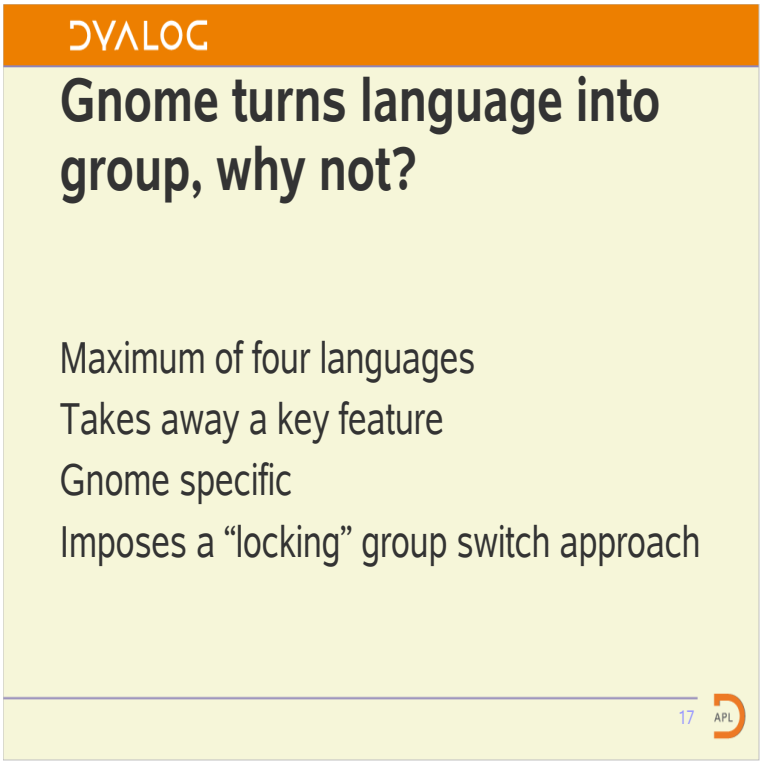
Sergey Udaltsov is the maintainer for XKB for Xorg and the maintainer for the XKB interfaces in Gnome.



By utilising the well defined ISO9995/XKB techniques for changing group. The use of the mouse or keyboard accelerators can be avoided.

Because the whole keyboard configuration is not changed on a language change there is minimal cost.

When the keyboard is changed a lot of information has to be transmitted between the X server and each of the X clients.

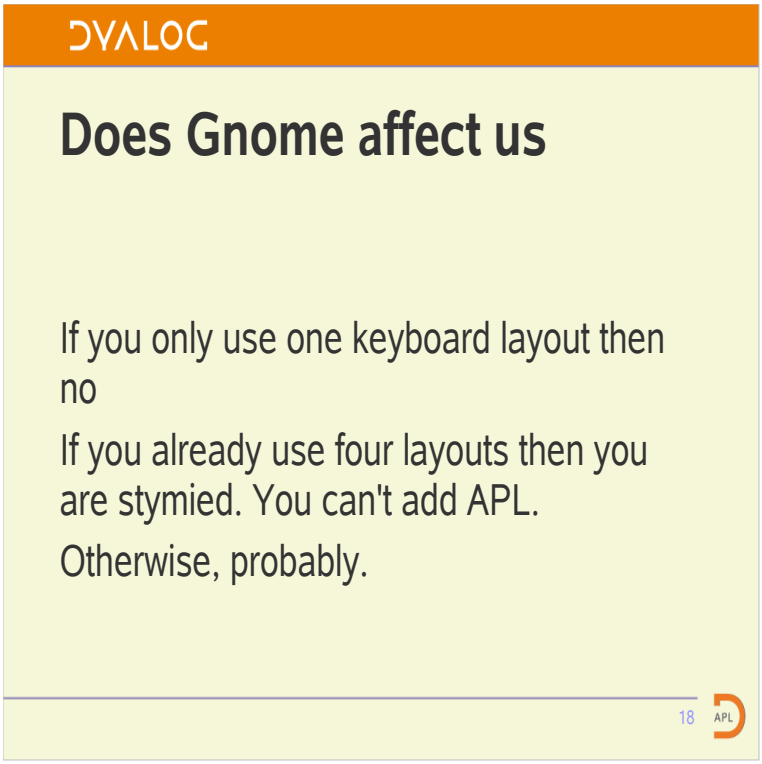


There is a brick wall with this approach at four groups. This limits you to four languages.

It restricts you from moving existing, and valid, XKB configurations from another X-server or desktop to Gnome.

You have to code your Xorg.conf (or equivalent) differently in order to accommodate the approach.

As soon as you have more than two languages the normal group switch technique is to use the “locking” approach mentioned earlier. This might not be a complete no-no. I think it is possible to define a latching key to go to a specific group and return to the current group when released.



Just one language? Then the same approach of using a second group just works.

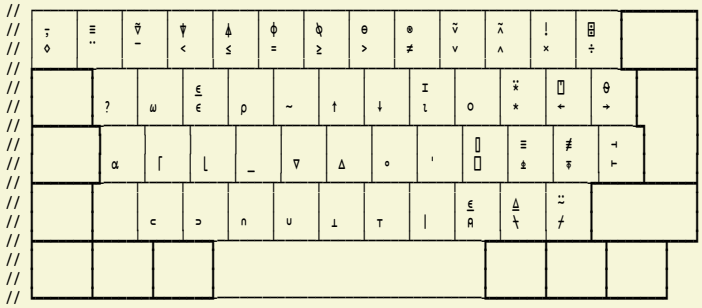
Polyglot? Use another desktop. They are not like football teams – pick one when you are seven and support it all your life. More like cars - choose one that suits your purpose at the time. You might respect and admire a Lotus Elise but it is no good for moving a ladder.

Two or three languages, this is quite common. I use US and GB. You (or me, probably me) will need to add the code for a group switch which latches a particular group and APL will need to be positioned so that it is in that group. As if I didn't have enough to do.

Enough of Gnome.

DYALOG

The APL keyboard



This is taken straight from the text comments in the file. The characters were all typed into vi after I had defined the keyboard.

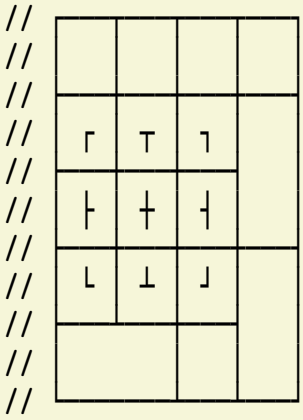
This is a conservative keyboard. Having abandoned the underscored alphabet and abandoned the idea of typing a third alphabet some of the keys seem to be strangely placed.

Why not place grade up on the same key as delta?

However, there is more.

DYALOG

Line graphics can be useful



However, I have only implemented one line thickness.

With the new unicode interpreter we could add others by using more levels.

DYALOG

How is it done?

```
key <AC09> {
    type[Group2] = "TWO_LEVEL",
    symbols[Group2] = [ U2395, U2337 ] // quad, squish
};
```


Back to our 'L' key.

The 'A' indicates the area of the keyboard.

The 'C' indicates the row labelling from the bottom up.

L is the 9th key on the row.

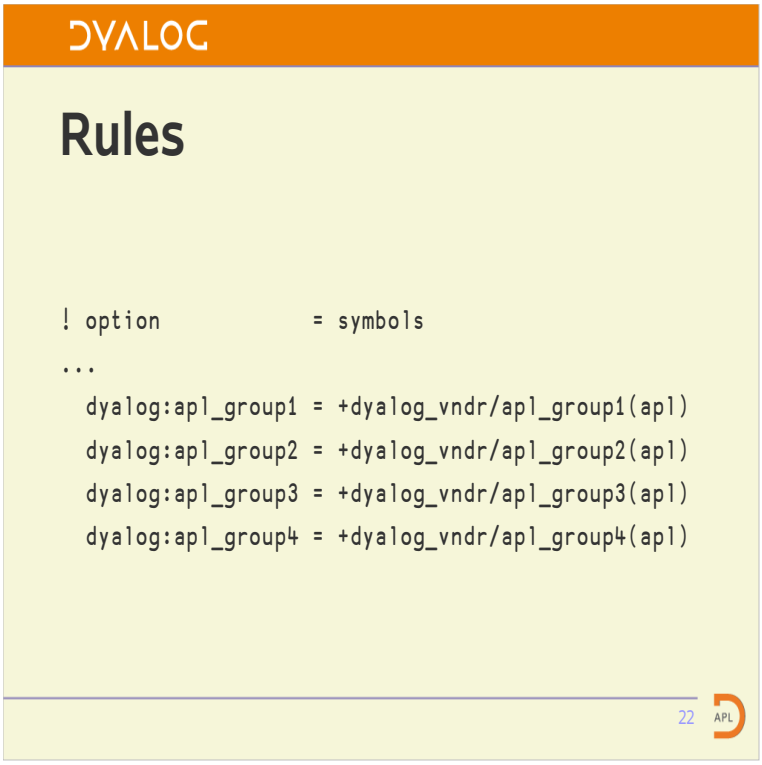
xkbprint is your friend. Except it's not fully unicode aware.

“TWO_LEVEL” means a basic key that can be shifted. CAPS LOCK has no affect.

Unicode code points are given for the generated characters.

Lots of boring repetition for various keys.

Done.



This is where I start to wave my hands about vaguely.

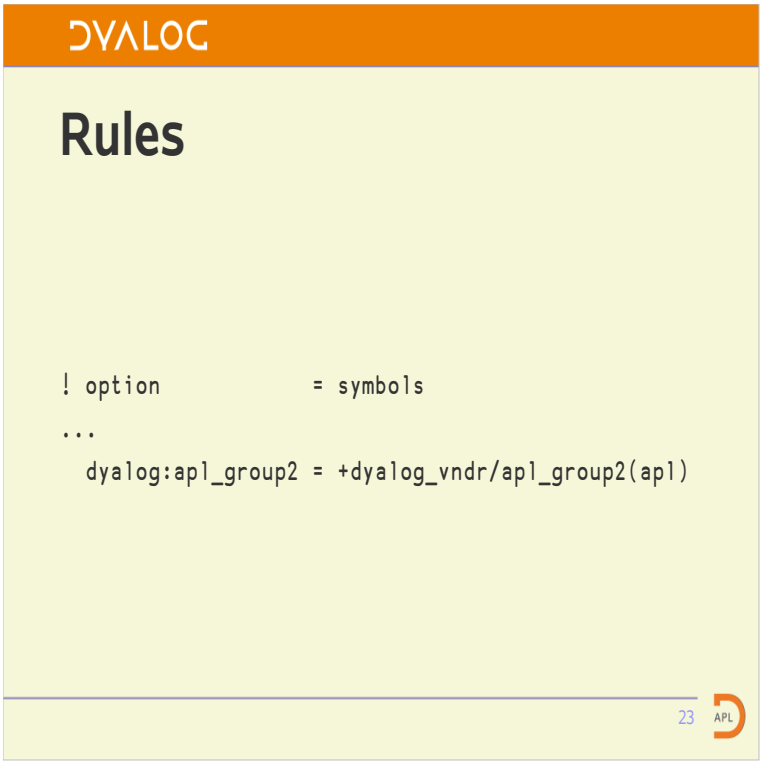
XKB builds upwards from

- geometry – a physical description
- keycodes – map scan codes to named keys
- types – specifies how keys produce levels
- symbols – what a key stroke produces

Xorg builds downwards from

- layout
- model
- options

The rules file ties the Xorg view to the XKB view. It allows users to chant incantations rather than seek understanding.



So this says: When you see the option dyalog:apl_group2 go and find the file symbols/dyalog_vndr/apl_group2.

In that file, find an xkb_symbols section called “apl”.

Add it to whatever you have so far.

xorg.conf

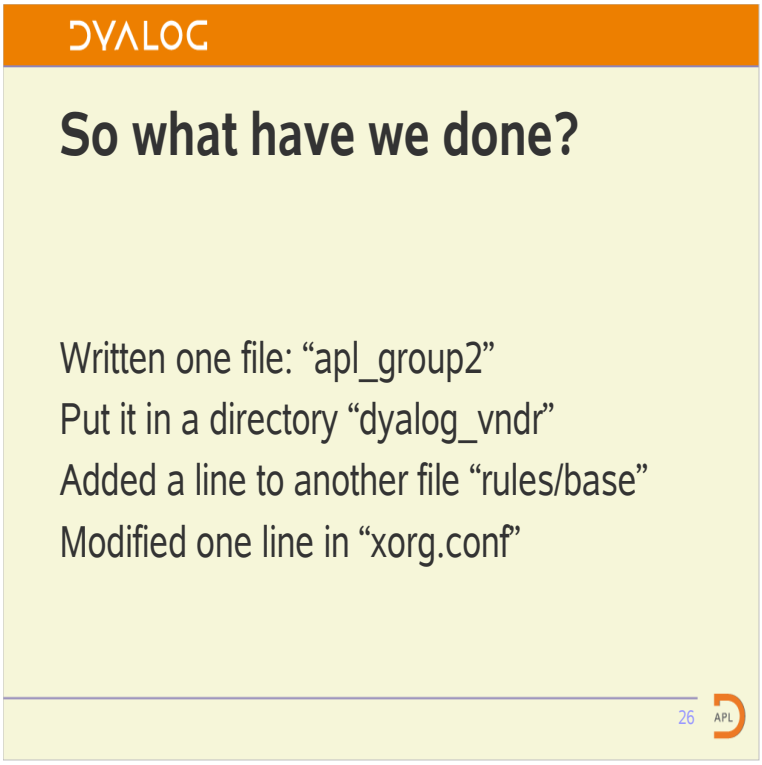
xorg.conf configures an X server
It is divided into sections.

DYALOG

```
Section "InputDevice"
  Driver      "kbd"
  Identifier  "Keyboard[0]"
  Option      "Protocol" "Standard"
  Option      "XkbLayout" "us,gb,dk"
  Option      "XkbTypes" "complete"
  Option      "XkbModel" "pc104"
  Option      "XkbGeometry" "pc"
  Option      "XkbCompat" "complete"
  Option      "XkbOptions" "dyalog:apl_group2,grp:lswitch"
  Option      "XkbRules" "xfree86"
  Option      "XkbVariant" "xfree86"
EndSection
```


This is where we invoke our option. It will add the APL symbols as group 2 to all of the layouts specified.

The other option specifies the group switch strategy. In this case the left alt key.

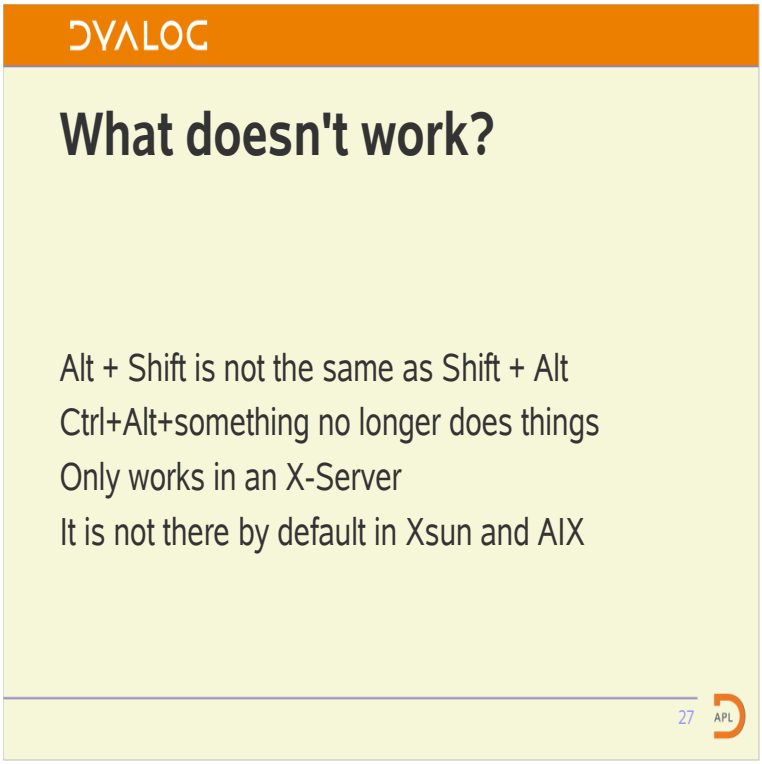


Actually, I added a README and a bash script to use “sed” to produce the files for the other groups.

I don't really know if creating a directory called “dyalog_vndr” is the right thing to do. However, it fits the style of the other things there.

Supporting the other groups actually makes this four lines.

The “Options” do all sorts of things – like swapping Ctrl with Caps Lock and choosing group switch strategies. Using it to add symbols seems fair.



I anticipate that this will cause some initial frustration but your fingers will rapidly learn.

I haven't found a way to use this positively yet. Maybe this is part of the answer to fitting 1024 characters on a single key. No – somehow I don't think so.

However, use the right side of the keyboard and they all work. So no loss of functionality.

You have to be running X. So you get no support at all in a raw Linux console.

If you are using Solaris or AIX as a desktop then getting XKB enabled will require some work – I don't know how much yet.

The keyboard can now talk to the application in our language.

Note that the other direction is still a little problematic.

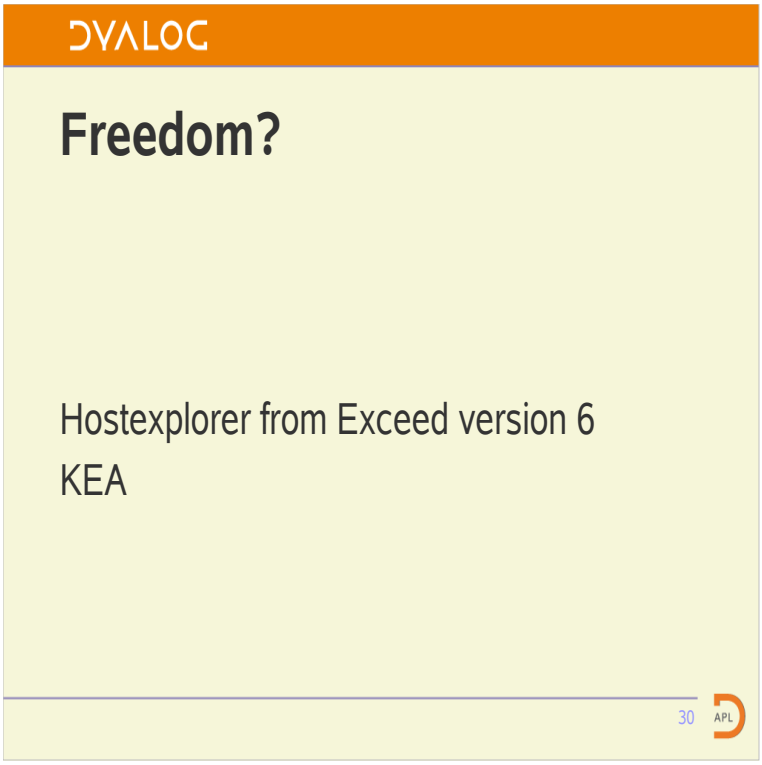
Most of the fonts on the system have not implemented the APL part of the unicode character set.

This is not as bad as it might be and it is getting better. There is Adrian Smith's APL385 font. There is “Misc Fixed” which is a single font. However, I don't think it is very pretty. There is a virtual font called “monospace”. Virtual fonts are a cascade of real or other virtual fonts. You get the character from the first font in the list that supplies it. Most of the APL characters are in “Cumberland AMT”, the rest seem to be in “Nimbus Mono L”.

Note that this doesn't just mean we can talk to Dyalog itself. We can talk to any application. We can type directly into OpenOffice. Directly into vi, Inkscape, Scribus ... the lot.

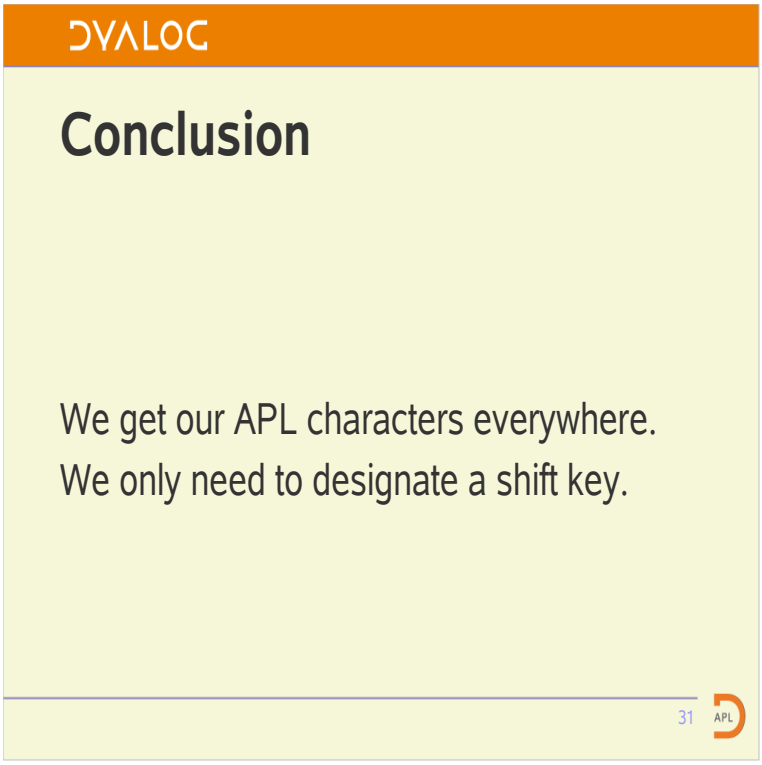
Font support for virtual fonts is a bit patchy. OpenOffice doesn't support them, which affected this presentation. However, gnome-terminal supports them. So I have taken to running gnome-terminal under KDE to run character based APL.

I have made input translates and output translates for UTF8 which can be used for all of our versions back to 6.2. You don't have to wait for 11.1 to use this work.



Some of our customers have persisted with the non GUI versions of Dyalog. They have been constrained to use one of two terminal emulators to achieve that. The main restriction was that the emulator had to support the downloadable font feature of the vt220. Versions of Exceed after 6 were broken in this respect.

Now they can use any terminal emulator that supports UTF8. If the desktop is Linux then that is fine and dandy. Windows now seems to be more of a problem. Running gnome-terminal on top of Xorg on top of Cygwin is quite a stack just to support a UTF8 terminal emulator.



To some extent I feel that I have put a lot of negative points into this presentation.

However, I am actually really pleased with what can be done. We don't have to click icons or use keyboard accelerators to bring up some IME. Our characters are just there ready for use.

Some people are going to be really pleased to just put the APL characters back on the Alt key.