Button Bar

COLLABORATORS			
	TITLE : Button Bar		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 19, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Button Bar

1.1 Button Bar - contents

```
BUTTON BAR  2.0


----------------------------------------

 I don't want to read your rotten manual...

 SHORT
 MOTIVATION
 REQUIREMENTS
 CLI USAGE
 USAGE

 CREDIT
 DISTRIBUTION
 BUGS AND SUGGESTIONS


 Technical details
```

1.2 technical details

```
 SYNTAX OF MENUFILE
 FIRST PART CONTROL SEQUENCES
 SECOND PART COMMANDS
 AREXX
```

1.3 nomanual

```
You don't have to. Just try typing in shell

     ButtonBar Example-menu
```

and experiment. It really should not be difficult.

1.4 short

Use this program to create your button interfaces for your work.
Each button executes a command specified by you.

1.5 motivation

When you work on something, you need to call different
programs with unusual parameters. It would be dirty to put these
calls into ToolsDaemon menu (which is the program I use mainly)
so you may want to create menu sitting on the screen (not like
menus needing right mouse button to be visible).
You can even use this program instead of ToolsDaemon, but
that's not the way I would do. For launching different sets of
programs from menu that does not change too much in time there's
hardly anything better than ToolsDaemon2.1.

1.6 requirements

ButtonBar requires "wbstart.library" V2.0+.

If you are not going to launch programs from WB, but only from CLI and
scripts, you don't actually need it. Most of WB programs can be run from
CLI anyway, but sometimes it is not convenient.

1.7 cli usage

ButtonBar MENUFILE/M,PORT/K

Menufile is the file containing definition of all buttons, which is
what text buttons show, what keys are assigned to buttons (if any) and
what each button actually do. If you don't supply menufile, "S:ButtonBar.menu"
will be taken instead. If you supply PORT argument, arexx port of that
name will be created. Notice that no arexx port is being created
without PORT argument given.

Example: ButtonBar Games-menu PORT="BB-GAMES"

1.8 usage

Mouse control is straightforward. You can also control program by cursor arrows and space or enter. In addition, you can also assign key to button. Esc key always quits ButtonBar. If Q key is not redefined by buttons, it also quits ButtonBar.

New in version 2.0
Editing of buttons and menus can be done by holding shift while clicking the mouse. You can control preferences like font, button width and many other options from menu.

1.9 syntax of menufile

All text coming before first menu-title is ignored. Control sequences (they begin with \ like in TeX) are the exception, even more they MUST come before first menu-title. That's why this doc can serve as menufile.

There are two logical parts of menu file - first is part before first menu-title, second is the rest.

First part may contain some ordinary text (like this doc) and also control sequences, which control behaviour of buttonbar. They will be listed later. Control sequences CANNOT be used in second part except sequences that will be executed upon button-press.

Organization of second part:

Buttons are organized into menus. You start menu definition by entering menu-title using ":menutitle". Next come your buttons. Button has first line for name, second for path of the program to be executed. Possible contents of second line:

. CLI-command argument1 argument2 ...

  command will be executed from its directory from CLI. E.g. for 'Games:Bleble/Bleblal aha' first 'cd Games:Bleble/' will be made and then 'Bleblal aha' executed.

. CLI-script argument1 argument2 ...

  Similar to above, except that script must have set flag S on. That's how ButtonBar tells, whether it is a script.

. <WB> WB-program

  WB-program will be run from WorkBench using information stored in WB-program.info

. SECOND-PART COMMAND - see later.

Line for name may be ended with |<key>. This way you can assign keys to buttons. Use _ to underscore letters in buttonname. If you use _ for underscoring, you don't need to use |. Character following _ will be automatically considered key shorcut.

Please note, that you MUST enter path for all programs. This means
that you cannot write 'run <>nil: work:something'. Instead, you
have to write 'c:run <>nil: work:something'.

In second part everything you write has importance, except comments.
Comments lines start with ;, # or /. Also empty line (no characters
including spaces) is not significant.

1.10 first part control sequences

\position <x>,<y>	coordinates of the bar.
\font <name>,<height>	used font. This file may or maynot end in .font
\buttonwidth <width>	width to be used by buttons. Default is 90.
\stablebar	this is a switch. It makes menubar not disappear when button executed.
\quitbutton	small quitbutton on the left will appear
\openmenu <i>	menu number {i} will be opened on the start. This sequence can be specified more times to open more menus.
\openmenu all	all menus will be opened on the start. but in exclusive mode (see below) finally right one menu will be open.
\exclusivemenus	Only one menu can be open at the time
\fixedopen	this option forces all menus to stay open once they were opened.
\opendown	sets prefered direction of menus. Default is up.
\runanddie	once some button is executed, buttonbar quits. This allows buttonbar to be used instead of multichoice (buttonbar has key support, which is not the case of multichoice.
\hotkey <key>	allows assigning standard commodities hotkey to the buttonbar. This doesnot work, if more buttonbars are running. (TO BE FIXED)
\workbench	this will force ButtonBar to be opened on Workbech screen. Otherwise ButtonBar opens on front-most screen.
\prefix <prefix>	sets string, which will be attached to commands startinf with \x. It can be thought of as macro

These control sequences are used WITHOUT the space between \ and
the rest. We put a space here so that these sequences don't actually
work!

1.11 second part commands

\quit	"quit" function.
\restart	"restart" function. Good for seeing effect of changed menufile.
\print <string>	this will print string to standard output. Using this command ButtonBar can behave like Commodore's RequestChoice, but with key support. Typical usage: set var=`buttonbar t:menu` where in menu, to each button only commands \print 1, \print 2 etc. are assigned. As you may have noticed, RequestChoice is ESSENCIALLY smaller than ButtonBar and has more elegant syntax. That's because it's single-purpose. But the missing key support is sad.
\x <command>	this command will be prefixed by <prefix> set in FIRST PART. The prefix is supposed to be command and will be followed by space before <command> is attached.
\title	this button cannot be executed. It will be in pushed state all the time so that it will look like menu titles.

1.12 arexx

If you don't supply REXXPORT name (see above), no REXXPORT will be created. ButtonBar understands following commands:

```
SLEEP   - ButtonBar goes to sleep, but stays in memory.
WAKEUP  - ButtonBar appears after being asleep.
QUIT    - ButtonBar quits and leaves all memory.
RESTART - ButtonBar closes all menus and rereads the menufile.
```

1.13 credit

Created by Daniel Polansky. I am a 4-5th year student of faculty of informatics on Masaryk University in Brno, Czechia. (Some of you maybe recognize this land as Czech Republic or even Czechoslovakia).
This program is rewrite of StartBar, which is PD. This program is much more configurable than StartBar. I appreciate help from guys from Blitzlist mailing list.

1.14 distribution

ButtonBar 2.0 is public domain. You can do anything you want with the software
except copyrighting it yourself. You can sell it, if you please, but you cannot
prevent your buyers from freely distributing it.

You are encouradged to distribute ButtonBar 2.0 with sources.

1.15 bugs and suggestions

Send any bugs and suggestions to: xpolansk@fi.muni.cz

I am not going to make any improvements in Blitz myself, just fixes if
you request some.

I think it would be great to rewrite Button Bar to GCC or why not straight
to G++ (GNU C++ compiler). Those Geek Gadgets guys did really great work,
don't you think.
Also rewriting to AmigaE might not be bad, but people, hook on C++!