.in 8.
.rm 72
.ce 6
Windows For C99

Edition 1.0 : 87/09/01

Software and Documentation written by
Tom 'c' Bentley

The c99 Windows library allows the 'c' programmer to use windows inside of his/her programs. To use c99 Windows the programmer simply opens windows and puts or gets information from them. Other features such as a message line, command bar and pop-up menus allow the programmer to make his programs very user friendly and pleasing to the eye.

To make the use of c99 Windows simple most of the input and output routines were modeled after existing 'c' functions, for example puts would be replaced w_puts to print strings to a window. Using this approach will allow you to change existing programs to utilize c99 Windows very quickly with a minimal amount of code change. Another nice feature of c99 Windows is. its' 80 column virtual display buffer, which will allow you to display up to 80 columns of information in the window, of course you will still have to flip through the virtual buffer using one of the supplied functions to see all of the information..
.
Since any type of windowing system takes a considerable amount of memory to run I have focused on creating a windowing library that has minimal memory usage, functionality and speed. Therefore there is very little to no error checking of parameters when calling any function so be careful..

There is no charge for this library, all I ask is if you develop any application using this library please send me a copy. If you have any problems or suggestions please send correspondence to:.

Tom Bentley
5662 Gordon St.
Box 346
Osgoode, Ontario
Canada K0A 2W0

.bp 2.

.ce
FIRST THINGS FIRST

The first thing you do is make a backup of the Windows library. Second build the object modules, to do this do the following:

1) Compile and assemble WINDOWSC to be WINDOWS.
2) Compile and assemble WCMDC to be WCMD.
3) Compile and assemble WCMDBARC to be WCMDBAR.
4) Compile and assemble WPOPUPC to be WPOPUP.
5) Compile and assemble WMSGC to be WMSG.
6) Compile and assemble WGETSC to be WGETS.
7) Compile and assemble WINDOWSG to be WINDOWSGO.
8) Compile and assemble WINDOWS1G to be WINDOWS1GO.
9) Compile and assemble WINDOWS2G to be WINDOWS2GO.
10) Compile and assemble WINDOWS3G to be WINDOWS3GO.

Compiler requirements: c99 version 2.1

The object file WINDOWS contain the kernal routines for c99 Windows and is always required when using c99 Windows. The other files are required when using other features of c99 Windows; WCMD is used to resize the window or scroll the virtual buffer, WCMDBAR is used when the command bar option is required, WPOPUP is used when you wish to use popup option windows, WMSG is required if you want to use the message line and WGETS is required if you want to get input from a window..

How to load the Libraries:

Using option 3 Load and Run or my program 'CLOAD' load the
following modules in the specified order:

  DSK1.C99PFI
  DSK1.<your object module(s)>
  DSK1.GRF1 (required)
  DSK1.WINDOWS (required)
  DSK1.WCMD (if required)
  DSK1.WCMDBAR (if required)
  DSK1.WPOPUP (if required)
  DSK1.WMSG (if required)
  DSK1.WGETS (if required)
  DSK1.CSUP (required)
  DSK1.C99PFF
  DSK1.WINDOWSGO (contains globals must be placed here!)
   or WINDOWS1GO/WINDOWS2GO/WINDOWS3GO (broken up WINDOWSGO)

Things for the future:

If this library is received well by c99 users I will continue
adding and improving it, so please send me your comments..
.bp
.ce
Table of Functions and Variables

.nf

Function or Variable   Page #

.bp
.ce
LIBRARY FUNCTIONS

.nf
----
 buf_col   variable
   Contains the starting column to display in window.

.fi

Calling Sequence:

```
extern int buf_col;
```

Description:

This variable is used to control which virtual buffer column is
the first column to be displayed within the current window..

.
.
Example:.
.
/* Program for buf_col example */.
.
#include "dsk1.windowsh".
.
extern int buf_col;

main().
  {.
    w_open("Windows");.
    w_puts("This is an example of how to use buf_col");.
    buf_col = 5; /* start display at 'is' */.
    w_flush();   /* flush it to force redisplay */.
    w_close();.
  }.
.bp.
.nf
---
 fillb   function
   Fill byte array with specified value.

.fi

Calling Sequence:

#include "dsk1.windowsh"

char *string, c;
.br
int   n;

fillb(string,c,n);

where string is the address to fill.
      c   is the fill character.
      n   is the number of c to fill into string.

Description:

Use fillb to fill a character value into some string array.

Returns:

Nothing

Object File Location:

WINDOWS

Example:

/* Program for fillb function */

#include "dsk1.windowsh"

main()
  {
    char string[81];

    fillb(string,'\0',81); /* fill string with nulls */
     .
     .
  }
.bp
.nf
---
  strlen   function
   Determine the length of a character string.

.fi

Calling Sequence:

```
#include "dsk1.windowsh"

char *string;
.br
int   siz;

siz = strlen(string);
```

where string is the pointer to the string to inspect.
siz receives the result of the count.

Description:

Use strlen to determine the length of a null terminated string.

Returns:

Returns the length of the string, in bytes.

Object File Location:

WINDOWS

Example:

```
/* Program of strlen function */

#include "dsk1.windowsh"

main()
  {
    char *string;
    int   len;

    string = "This is a string";
    len = strlen(string);
     .
     .
  }
```

.bp
.nf

---

use_vbuf variable

Flag specifiying whether to use the virtual buffer.

.fi

Calling Sequence:

```
extern int use_vbuf;
```

Description:

This variable is used to control whether information is written into the 80 column virtual buffer. Set it to 1 to use the virtual buffer, otherwise set it to 0..

Example:

```
/* Program for use_vbuf variable */

#include "dsk1.windowsh"

extern int use_vbuf;

main().
  {.
    w_open("Windows");.
    use_vbuf = 0; /* don't use virtual buffer */.
    w_puts("Not using virtual buffer");.
    w_close();.
```

```
  }.
.bp
.nf
------------------------------------------------------------------
 w_box                                                    function
   Draw or erase a box on the screen.
------------------------------------------------------------------
.fi

Calling Sequence:

#include "dsk1.windowsh"

int top_row, top_left_col, bottom_row;
.br
int bottom_right_col, clear;

w_box(top_row,top_left_col,bottom_row,bottom_right_col,clear);

where top_row is the top row of the box.
      top_left_col is the top left column of the box.
      bottom_row is the bottom row of the box.
      bottom_right_col is the bottom right column of the box.
      clear specifies whether the box is drawn or erased.


Description:

Use  w_box  to draw or erase a box on the screen.  A box will be
drawn if clear is false otherwise it will  be  erased  from  the
screen.


Returns:

Nothing


Object File Location:

WINDOWS

Example:

/* Program for w_box function */

#include "dsk1.windowsh"

main()
  {
    w_init();
    w_box(2,1,23,40,0); /* draw a box */
    w_box(2,1,23,40,1); /* erase the box just drawn */
     .
     .
  }
.bp
.nf
------------------------------------------------------------------
 w_clear                                                  function
   Clears the current window display
------------------------------------------------------------------
.fi

Calling Sequence:

#include "dsk1.windowsh"

w_clear();


Description:

Use w_clear if you wish to clear the current window.


Returns:
```

Nothing

Object File Locatation:

WINDOWS

Example:

```
/* Program for the w_clear function */

#include "windowsh"

main()
  {
    w_init();
    w_open("Windows");
    w_puts("Lines 1\nLine 2\n);
    w_clear();
  }
```

.bp
.nf

---

w_close function
Closes the current window.

.fi

Calling Sequence:

```
#include "dsk1.windowsh"

w_close();
```

Description:

Use w_close to close the current window. When the window is closed it also clears it from the screen.

Returns:

Nothing

Oject File Location:

WINDOWS

Example:

```
/* Program for the w_close function */

#include "dsk1.windowsh"

main()
  {
    w_init();
    w_open("Windows");
     .
     .
    w_close();
  }
```

.bp
.nf

---

w_clrbuf function
Initializes the virtual buffer to nulls.

.fi

Calling Sequence:

```
#include "dsk1.windowsh"
```

```
w_clrbuf();
```

Description:

Use w_clrbuf to initialize the virtual buffer to nulls. The virtual buffer consists of 20 rows by 80 columns.

Returns:

Nothing

Object File Location:

WINDOWS

Example:

```
/* Program for the w_clrbuf function */

#include "dsk1.windowsh"

main()
  {
    w_init();
     .
     .
    w_clrbuf();
     .
     .
  }
.bp
.nf
-------------------------------------------------------------------
 w_cmd                                                    function
   Resize window or flip virtual buffer in window.
-------------------------------------------------------------------
.fi
```

Calling Sequence:

```
#include "dsk1.windowsh"
.br
extern w_cmd();

char c;

c = w_cmd(c);
```

where c is the entered command, or the command to execute.

Description:

Use w_cmd if you wish to resize the current window or scan through the virtual buffer. If you pass a non NULL character value to w_cmd then this value will be used to perform the action instead of getting input from the keyboard. Control S,D,E or X will resize the window using the specified direction. Function 5 (next window) will scan through the virtual buffer and redisplay it to the window..

Returns:

Returns the specified command as a character.

Object File Location:

WCMD

Example:

```
/* Program for the w_cmd function */

#include "dsk1.windowsh"
.br
extern w_cmd();

main()
  {
    w_init();
    w_open("Window");
    w_puts("This is an example of how to use w_cmd");
    while(1)
        w_cmd();
  }
.bp
.nf
-----------------------------------------------------------------
 w_cmdbar                                                function
   Display command bar with options.
-----------------------------------------------------------------
.fi

Calling Sequence:

#include "dsk1.windowsh"
.br
extern w_cmdbar();

char *options;
.br
int   n;

n = w_cmdbar(options)

where options is the string containing command options.
      n       is the option selected.


Description:

Use w_cmdbar to display a command bar on line 1 with the
specified options.  The option string is a stream of characters
with each option separated with a newline (\n).  There may be up
to 20 options on the command bar but the total bar length may
not exceed 40 characters, the newlines are not counted in the
length.  The first option is numbered option 1 and will be the
first option on the left of the command bar (it will be
inverted).  To select a specific option use the arrow keys
(function S and D) until you are on the option you wish to
choose and press enter to choose it.  If you do not wish to
select anything then press function 9 (BACK)..


Returns:

The option you selected from 1 to the maximum options you have
specified or 0 if function BACK was entered.  If w_raw is set to
1 then any character pressed will be returned, see w_raw for
more detail..


Object File Location:

WCMDBAR

.bp
Example:

/* Program for the w_cmdbar function */

#include "dsk1.windowh"
.br
extern w_cmdbar();

main()
  {
    int opt;
```

```
    w_init();
    opt = w_cmdbar(" option 1 \n option 2 \n");
     .
     .
  }
.bp
.nf
-------------------------------------------------------------------
 w_drum                                                   variable
   Contains pointers to access virtual buffer.
-------------------------------------------------------------------
.fi

Calling Sequence:

extern int w_drum[];


Description:

This variable contains byte pointers to the virtual buffer
lines.  Using these pointers you may access or alter the
contents of the virtual buffer at any time.  This variable
consists of 20 buffer pointers, each containing a pointer to its
respective buffer line..

Example:.
.
#include "dsk1.windowsh".
.
extern int w_drum[];.
.
main().
  {.
    char *lin;  /* used for cast */.
    char *lin2;
.
    w_open("Windows");
    w_puts("Line 1.");.
    lin = w_drum[0]; /* pointer to 1st line */.
    lin2 = w_drum[1]; /* pointer to 2nd line */.
    strcpy(lin2,lin); /* copy line 1 to line 2 */.
    w_close();
  }
.bp
.nf
-------------------------------------------------------------------
 w_flush                                                  function
   Flushes the virtual buffer to the current window.
-------------------------------------------------------------------
.fi

Calling Sequence:

#include "dsk1.windowsh"

w_flush();


Description:

Use w_flush to flush the 80 column virtual buffer to the current
window.


Returns:

Nothing


Object File Location:

WINDOWS


Example:

/* Program for the w_flush function */
```

```
#include "dsk1.windowsh"

main()
  {
    w_init();
    w_open("Window");
    w_puts("Information\n");
    w_open("New Window);
    w_flush(); /* flush previous windows' buffer to this one */
  }
.....................
```