Getting terminal size in c for windows?

Asked 12 years, 5 months ago Modified 5 years, 7 months ago Viewed 37k times

31

How to check ymax and xmax in a console window, under Windows, using plain c?

There is this piece of code for linux:

```
#include <stdio.h>
#include <sys/ioctl.h>
int main (void)
{
    struct winsize max;
    ioctl(0, TIOCGWINSZ , &max);
    printf ("lines %d\n", max.ws_row);
    printf ("columns %d\n", max.ws_col);
}
```

Now I wonder how can I do the same for windows. I tried `winioctl.h` but it does not define `struct winsize` nor any other with a similar name.

Any tips? Thanks.

PS. In linux you also can find the console size using `getenv("LINES");` . Is there a similar variable under windows?

PPS. Also, there is always `ncurses.h` , that I suppose work both systems, but I'm avoiding it because of conflicts with other libraries I have.

PPPS. This question here Getting terminal width in C? has a lot of tips, so no need to repeat that.

c console size windows-console

Share Edit Follow

edited May 23, 2017 at 11:54
Community Bot
1 1

asked Jul 25, 2011 at 5:53
DrBeco
11.4k 10 59 77

3 Answers

Sorted by: Highest score (default)

53

This prints the size of the console, not the buffer:

```
#include <windows.h>

int main(int argc, char *argv[]) {
```

```
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    int columns, rows;

    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbi);
    columns = csbi.srWindow.Right - csbi.srWindow.Left + 1;
    rows = csbi.srWindow.Bottom - csbi.srWindow.Top + 1;

    printf("columns: %d\n", columns);
    printf("rows: %d\n", rows);
    return 0;
}
```

This code works because `srWindow` "contains the console screen buffer coordinates of the upper-left and lower-right corners of the display window", and the `SMALL_RECT` structure "specify the rows and columns of screen-buffer character cells" according to MSDN. I subtracted the parallel sides to get the size of the console window. Since I got `1` less than the actual value, I added one.

Share Edit Follow

edited Oct 1, 2012 at 16:31

answered Sep 28, 2012 at 15:14

quantum
3,752 30 51

I'm short on time to test it, but it looks like this is the right answer to the question. Would you mind point out the difference between this answer and the one currently accepted that shows the buffer size? Thanks! – DrBeco Sep 30, 2012 at 20:07

1 @DrBeco In most cases, the buffer size from your answer, and the display window mentioned in this answer have the same width, although this is not required. However, as you note the buffer and windows can and often do have different heights - this is what provides for the ability to scroll back up a Windows console session. I find it easiest to think in terms of a large buffer, with a smaller viewport (display) window that can look at any section of the main buffer. – dgnuff Mar 18, 2016 at 21:42

Exactly what I was looking for. +1 – Xam Mar 30, 2018 at 23:18

1 Or you could use CONSOLE_SCREEN_BUFFER_INFO.dwSize.X and .Y for short. – RhetoricalRuvim Aug 1, 2019 at 18:31

2 If STDOUT is redirected to a file, then the `GetConsoleScreenBufferInfo()` call returns `FALSE`, so that should be tested. – Craig McQueen Sep 29, 2020 at 5:26

9

(Partial answer)

This code:

```
CONSOLE_SCREEN_BUFFER_INFO csbi;
int ret;
ret = GetConsoleScreenBufferInfo(GetStdHandle( STD_OUTPUT_HANDLE ),&csbi);
if(ret)
{
    printf("Console Buffer Width: %d\n", csbi.dwSize.X);
    printf("Console Buffer Height: %d\n", csbi.dwSize.Y);
}
```

Gives the size of the buffer. The only problem is that `dwSize.Y` is not really the size of the screen (300 here instead of 25 lines). But `dwSize.X` matches the column's number. Needs only `windows.h` to work.

Share Edit Follow

answered Jul 25, 2011 at 12:06

DrBeco
11.4k 10 59 77

4

The below two functions will get the window size somewhat more directly.

Note that I found, using gcc, that neither this approach nor GetConsoleScreenBufferInfo works if the program is piped. That is somewhat of a pain as for/f then does not work either. Apparently the screen data is not available in a pipe.

Um, the above remark is of course enormously stupid. ;) It is STDOUT that is not the screen in a pipe! That does mean I prefer using STD_ERROR_HANDLE above STD_OUTPUT_HANDLE. I am far less likely to direct standard error away from the screen than standard output.

```
typedef struct _CONSOLE_FONT_INFO {
  DWORD nFont;
  COORD dwFontSize;
} CONSOLE_FONT_INFO, *PCONSOLE_FONT_INFO;

BOOL WINAPI GetCurrentConsoleFont(
   HANDLE              hConsoleOutput,
   BOOL                bMaximumWindow,
   PCONSOLE_FONT_INFO lpConsoleCurrentFont
);

/* Get the window width */
int getww_(void)
{
    CONSOLE_FONT_INFO info;
    GetCurrentConsoleFont(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &info);
    return info.dwFontSize.X;
}

/* Get the window height */
int getwh_(void)
{
    CONSOLE_FONT_INFO info;
    GetCurrentConsoleFont(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &info);
    return info.dwFontSize.Y;
}
```

Share Edit Follow

edited May 17, 2018 at 17:01

answered May 17, 2018 at 15:44

Leon van Dommelen
101 1 5

Since this is a snippet, I add #include <windows.h> and a main() to print getww() and getwh(). When I run that, I get the size of the FONT being used - 12x7 - NOT the console size. Am I doing something wrong? – cniggeler Apr 29, 2021 at 13:31

1 +1 for suggesting STD_ERROR_HANDLE to get @quantum's code to work if you use it in a script or otherwise need to pipe the output. – cniggeler Apr 29, 2021 at 13:38