Bring Colors to the Windows Console

André Burgaud

2009-05-10

Contents

The code in this article demonstrates how to write foreground and background colored text in a Windows Console using Python and ctypes.

Custom Implementation with ctypes

ctypes is a foreign function interface (FFI) library for Python written by Thomas Heller. It has been part of the Python standard library since Python 2.5.

FFIs expose mechanism for a host language to access functions defined in a guest language. For example, in Python, ctypes implements an interface allowing access to functions defined in a C library. These libraries may have different formats depending on the operating system:

- DLL on Windows
- .so Libraries on Unix-like systems

Often overlooked on Windows, command line programs can be very powerful. The ability to change the color and intensity of the text in the terminal may improve the experience of command line users.

For example, UPX (the Ultimate Packer for eXecutables) takes advantage of this concept and it inspired me to introduce this feature into my Python command line programs. When you issue the `upx --help` in the Windows Console, you obtain the following output:

The UPX help command output shows a description section colored in green and highlights the headers of each following sections in yellow.

Coloring output text is not available in the Python standard library, but there are tools to implement this feature are. On Windows, using `ctypes` allows to call the appropriate Win32 functions, in particular SetConsoleTextAttribute() and GetConsoleScreenBufferInfo().

Here is a code example showing a possible approach to wrap these Win32 functions like `SetConsoleTextAttribute()` and the required data structures like CONSOLE_SCREEN_BUFFER_INFO:

```
# color.py
"""
Colors text in console mode application (win32).
Uses ctypes and Win32 methods SetConsoleTextAttribute and
GetConsoleScreenBufferInfo.
"""

import ctypes
```

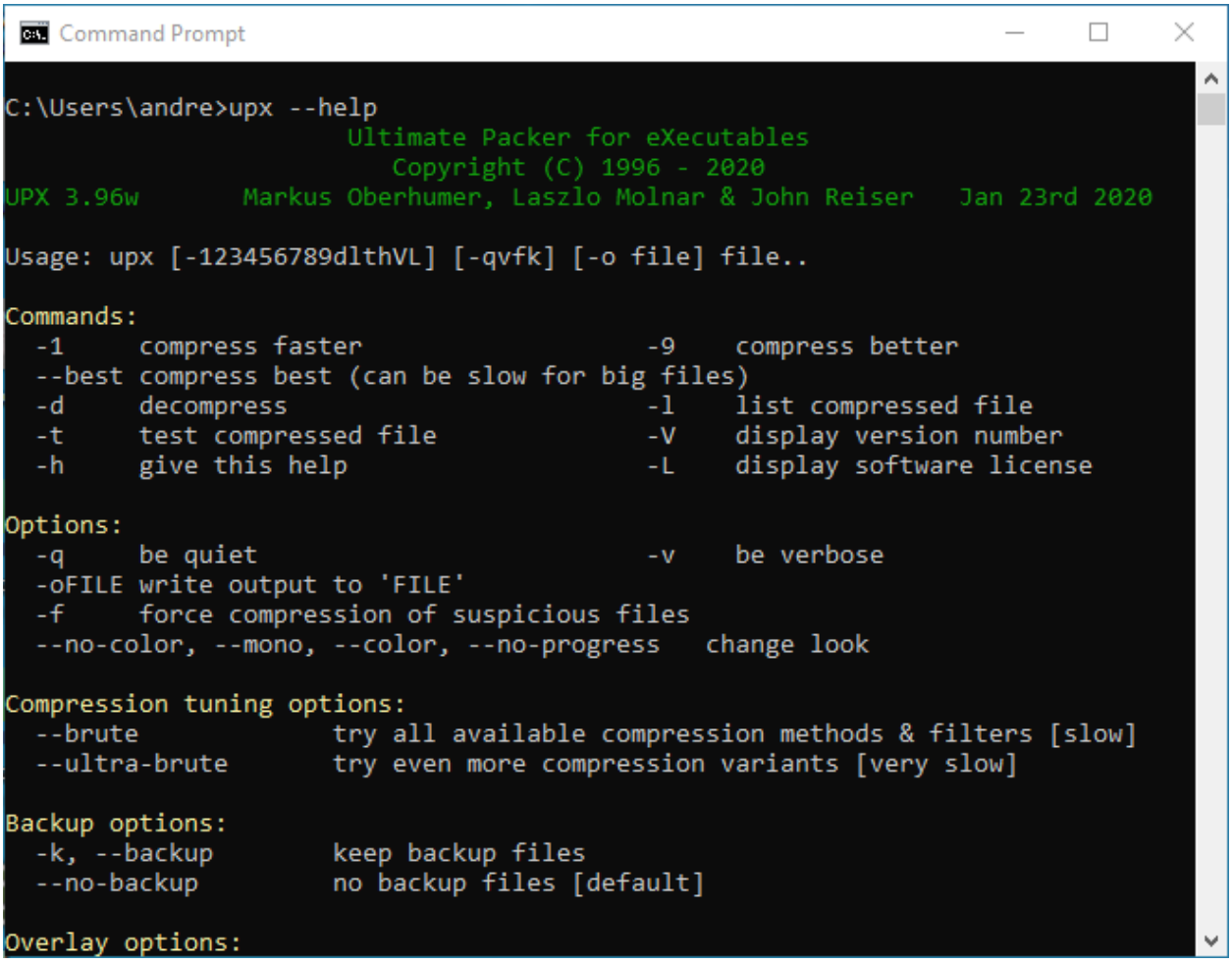

Figure 1: UPX Screenshot

```
SHORT = ctypes.c_short
WORD = ctypes.c_ushort


class COORD(ctypes.Structure):
    """struct in wincon.h."""

    _fields_ = [("X", SHORT), ("Y", SHORT)]


class SMALL_RECT(ctypes.Structure):
    """struct in wincon.h."""

    _fields_ = [("Left", SHORT), ("Top", SHORT), ("Right", SHORT), ("Bottom", SHORT)]


class CONSOLE_SCREEN_BUFFER_INFO(ctypes.Structure):
    """struct in wincon.h."""

    _fields_ = [
        ("dwSize", COORD),
        ("dwCursorPosition", COORD),
        ("wAttributes", WORD),
        ("srWindow", SMALL_RECT),
        ("dwMaximumWindowSize", COORD),
    ]


# winbase.h
STD_INPUT_HANDLE = -10
STD_OUTPUT_HANDLE = -11
STD_ERROR_HANDLE = -12

# wincon.h
FOREGROUND_BLACK = 0x0000
FOREGROUND_BLUE = 0x0001
FOREGROUND_GREEN = 0x0002
FOREGROUND_CYAN = 0x0003
FOREGROUND_RED = 0x0004
FOREGROUND_MAGENTA = 0x0005
FOREGROUND_YELLOW = 0x0006
FOREGROUND_GREY = 0x0007
FOREGROUND_INTENSITY = 0x0008  # foreground color is intensified.

BACKGROUND_BLACK = 0x0000
BACKGROUND_BLUE = 0x0010
BACKGROUND_GREEN = 0x0020
BACKGROUND_CYAN = 0x0030
BACKGROUND_RED = 0x0040
BACKGROUND_MAGENTA = 0x0050
BACKGROUND_YELLOW = 0x0060
BACKGROUND_GREY = 0x0070
BACKGROUND_INTENSITY = 0x0080  # background color is intensified.
```

```
stdout_handle = ctypes.windll.kernel32.GetStdHandle(STD_OUTPUT_HANDLE)
SetConsoleTextAttribute = ctypes.windll.kernel32.SetConsoleTextAttribute
GetConsoleScreenBufferInfo = ctypes.windll.kernel32.GetConsoleScreenBufferInfo


def get_text_attr() -> WORD:
    """Returns the character attributes (colors) of the console screen
    buffer."""
    csbi = CONSOLE_SCREEN_BUFFER_INFO()
    GetConsoleScreenBufferInfo(stdout_handle, ctypes.byref(csbi))
    return csbi.wAttributes


def get_distinct_attr() -> (WORD, WORD, WORD, WORD):
    """Returns a tuple with 4 values: foreground color, foreground intensity,
    background color, and background intensity"""
    attr = get_text_attr()
    return (
        attr & FOREGROUND_GREY,
        attr & FOREGROUND_INTENSITY,
        attr & BACKGROUND_GREY,
        attr & BACKGROUND_INTENSITY,
    )


def set_text_attr(color: int) -> None:
    """Sets the character attributes (colors) of the console screen
    buffer. Color is a combination of foreground and background color,
    foreground and background intensity."""
    SetConsoleTextAttribute(stdout_handle, color)
```

You can colorized text in the Windows Console by importing `color` in your Python code:

```
# color_console.py
"""
Test module color. Requires Python 3.
"""


import sys
import color


def test():
    """Simple Python test for color."""
    default_colors = color.get_text_attr()  # Save default attributes to reset later
    _, _, bg_color, bg_intensity = color.get_distinct_attr()
    color.set_text_attr(
        bg_color | bg_intensity | color.FOREGROUND_BLUE | color.FOREGROUND_INTENSITY
    )
    print("============================================")
    color.set_text_attr(
        color.FOREGROUND_BLUE
        | color.BACKGROUND_GREY
```

```
        | color.FOREGROUND_INTENSITY
        | color.BACKGROUND_INTENSITY
    )
    print("And Now for Something", end=" ")
    sys.stdout.flush()  # Force writing first part of the line in blue
    color.set_text_attr(
        color.FOREGROUND_RED
        | color.BACKGROUND_GREY
        | color.FOREGROUND_INTENSITY
        | color.BACKGROUND_INTENSITY
    )
    print("Completely Different!")
    color.set_text_attr(
        bg_color | bg_intensity | color.FOREGROUND_RED | color.FOREGROUND_INTENSITY
    )
    print("============================================")
    color.set_text_attr(default_colors)  # Reset to default console attributes


if __name__ == "__main__":
    test()
```

You can execute `color_console.py` with the following command:

```
c:/> python color_console.py
```

You will obtain the following colorful output of And Now for Something Completely Different:

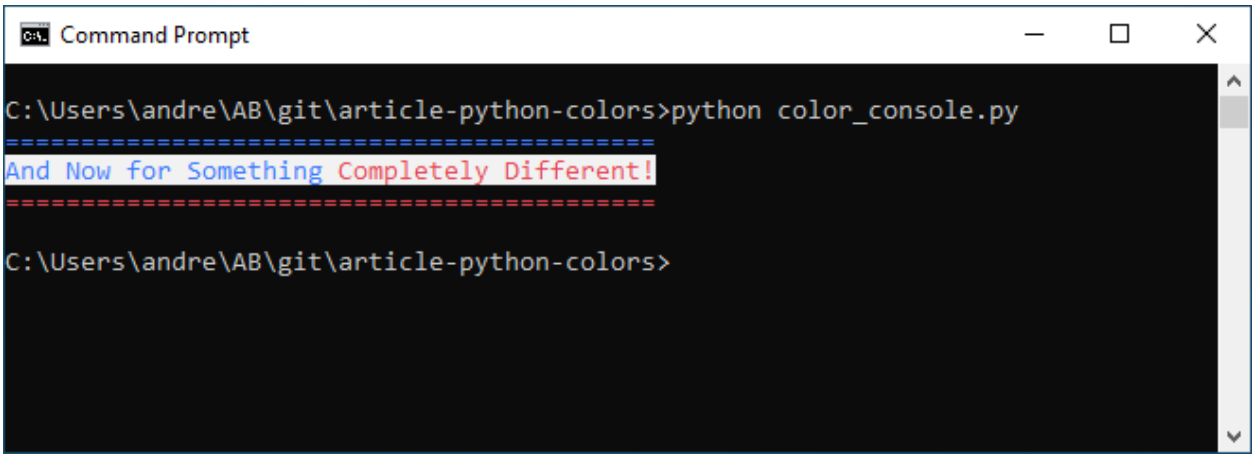

Figure 2: Windows Console with Colored Fonts

Related Work

The Python library colorama is a complete and cross-platform implementation of the feature described in this article.

Updates

- **03/07/2021**:
  - Removed the Python 2 examples
  - Added a section on related work (colorama)

- Added Python types
- Formatted code with black
- Updated screenshots
- Tested with Python 3.9.2

Source Code