

Forum:[Hack Forums](#)

Thread: [Best way to handle unicode / char conversions in Cpp?](#)

Hackers - December 22, 2023, 8:50 am

What is an elegant way to handle converting keycodes to standard keyboard layout labels/letters? This is an area I am struggling with due to number of exceptions and unique key combos and usage, so am wondering what the industry standard is. Specifically I'd be interested in something that also has localization for multi language/ keyboard layouts.

an4l 1st - December 22, 2023, 10:15 am

depends, is it for a keylogger? what method? getasynckeystate?

Hackers - December 22, 2023, 11:28 pm

(Dec 22, 2023 11:15 AM)Threat Actress Wrote: [\[CITE\]](#) depends, is it for a keylogger? what method? getasynckeystate?

Just in general - could be for logging all keys, only some keys, etc. Just curious what people have done to convert keycodes into readable chars. I know how to do it in other languages, but not sure what best practice is for C++. As for exact tools, what's your recommendation? I am starting with windows.h

an4l 1st - December 23, 2023, 5:41 am

(Dec 23, 2023 12:28 AM)nodes Wrote: [\[CITE\]](#) Just in general - could be for logging all keys, only some keys, etc. Just curious what people have done to convert keycodes into readable chars. I know how to do it in other languages, but not sure what best practice is for C++. As for exact tools, what's your recommendation? I am starting with windows.h

I guess you could use ToUnicodeEx, depending on how you are getting the keycodes...

something along theses lines:

```
Code #include <iostream>
#include <Windows.h>
```

```
int main() {
    // Get the current keyboard layout
    HKL keyboardLayout = GetKeyboardLayout(0);

    // Simulate a key press event
    INPUT input;
    input.type = INPUT_KEYBOARD;
    input.ki.wVk = VK_A; // Press the 'A' key
    input.ki.dwFlags = 0;
    SendInput(1, &input, sizeof(INPUT));

    // Sleep to give the system some time to process the input
    Sleep(100);

    // Simulate a key release event
    input.ki.dwFlags = KEYEVENTF_KEYUP;
    SendInput(1, &input, sizeof(INPUT));
}
```

```

// Get the keyboard state
BYTE keyboardState[256];
GetKeyboardState(keyboardState);

// Buffer to store the Unicode result
WCHAR result[2]; // Assuming a single Unicode character result

// Call ToUnicodeEx to translate the virtual key code to Unicode
int resultLength = ToUnicodeEx(VK_A, 0, keyboardState, result, 2, 0, keyboardLayout);

// Check if the translation was successful
if (resultLength > 0) {
    std::wcout << L"Unicode character: " << result[0] << std::endl;
} else {
    std::wcout << L"Translation failed." << std::endl;
}

return 0;
}

```

Hackers - December 23, 2023, 6:04 am

(Dec 23, 2023 06:41 AM)Threat Actress Wrote: [\[CITE\]](#) I guess you could use ToUnicodeEx, depending on how you are getting the keycodes...

something along theses lines:

```

Code #include <iostream>
#include <Windows.h>

int main() {
    // Get the current keyboard layout
    HKL keyboardLayout = GetKeyboardLayout(0);

    // Simulate a key press event
    INPUT input;
    input.type = INPUT_KEYBOARD;
    input.ki.wVk = VK_A; // Press the 'A' key
    input.ki.dwFlags = 0;
    SendInput(1, &input, sizeof(INPUT));

    // Sleep to give the system some time to process the input
    Sleep(100);

    // Simulate a key release event
    input.ki.dwFlags = KEYEVENTF_KEYUP;
    SendInput(1, &input, sizeof(INPUT));

    // Get the keyboard state
    BYTE keyboardState[256];
    GetKeyboardState(keyboardState);

    // Buffer to store the Unicode result

```

```
WCHAR result[2]; // Assuming a single Unicode character result
```

```
// Call ToUnicodeEx to translate the virtual key code to Unicode
```

```
int resultLength = ToUnicodeEx(VK_A, 0, keyboardState, result, 2, 0, keyboardLayout);
```

```
// Check if the translation was successful
```

```
if (resultLength > 0) {
```

```
    std::wcout << L"Unicode character: " << result[0] << std::endl;
```

```
} else {
```

```
    std::wcout << L"Translation failed." << std::endl;
```

```
}
```

```
    return 0;
```

```
}
```

I guess I'm a little confused by your comment about how I'm getting the codes because keyboard keycodes are fairly standardized in other languages where a key with ID 37 is 37 across multiple programming languages. I'm guessing that's not always true, but can confirm the codes in my current C++ program are just standard integers per key, matching other langs, but my conversion function is failing. I'll try out your translation to check later.