
Subject: access to raw command line parameters
Posted by [dr_jumba](#) on Tue, 04 Jul 2006 16:21:30 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi,

Is there exist an easy way to access to command line parameters passed to GUI_APP?

What I need is to pass these parameters to 3rd party library init function.

E.g.

```
SuperLibInit(argc, argv);
```

Thanks.

Subject: Re: access to raw command line parameters
Posted by [fallingdutch](#) on Wed, 05 Jul 2006 07:03:48 GMT
[View Forum Message](#) <> [Reply to Message](#)

dr_jumba wrote on Tue, 04 July 2006 18:21Is there exist an easy way to access to command line parameters passed to GUI_APP?

yes, as far as i can see you get them by calling "CommandLine()" it returns a Vector of Strings (Vector<String> &)

dr_jumba wrote on Tue, 04 July 2006 18:21What I need is to pass these parameters to 3rd party library init function.

E.g.

```
SuperLibInit(argc, argv);
```

I am not sure, but take a look at dli (Core/dli.h, Core/Dli.cpp) in the "Win32 support" section of Core.

As far as i know you can even load .so at *nix OS, using these files, but not sure.

If you have a solution for this please poste it, working on it, too

Bas

Subject: Re: access to raw command line parameters
Posted by [unodgs](#) on Wed, 05 Jul 2006 07:04:44 GMT
[View Forum Message](#) <> [Reply to Message](#)

dr_jumba wrote on Tue, 04 July 2006 12:21Hi,

Is there exist an easy way to access to command line parameters passed to GUI_APP?

What I need is to pass these parameters to 3rd party library init function.

E.g.

```
SuperLiblNit(argc, argv);
```

Thanks.

It seems there is no way to access that variables directly, but you can get command line parameters into string vector using CommnadLine() function:

```
const Vector<String> &cmd = CommandLine();
int cnt = cmd.GetCount();
const char **argv = new const char*[cnt];
```

```
for(int i = 0; i < cnt; i++)
    argv[i] = cmd[i];
```

```
SuperLiblNit(cnt, argv);
```

Subject: Re: access to raw command line parameters

Posted by [Werner](#) on Sat, 08 Jul 2006 20:08:45 GMT

[View Forum Message](#) <> [Reply to Message](#)

unodgs wrote on Wed, 05 July 2006 09:04dr_jumba wrote on Tue, 04 July 2006 12:21Hi,

Is there exist an easy way to access to command line parameters passed to GUI_APP?

What I need is to pass these parameters to 3rd party library init function.

E.g.

```
SuperLiblNit(argc, argv);
```

Thanks.

It seems there is no way to access that variables directly, but you can get command line parameters into string vector using CommnadLine() function:

```
const Vector<String> &cmd = CommandLine();
int cnt = cmd.GetCount();
const char **argv = new const char*[cnt];
```

```
for(int i = 0; i < cnt; i++)
    argv[i] = cmd[i];
```

```
SuperLiblNit(cnt, argv);
```

If the above code doesn't help (it doesn't comply with the C++ standard) you might want to have snippets, including reference examples!)

Werner

Subject: Re: access to raw command line parameters
Posted by [unodgs](#) on Sat, 08 Jul 2006 20:32:21 GMT
[View Forum Message](#) <> [Reply to Message](#)

Werner wrote on Sat, 08 July 2006 16:08
If the above code doesn't help (it doesn't comply with the C++ standard)

May I know what's wrong with it??

Subject: Re: access to raw command line parameters
Posted by [Werner](#) on Sat, 08 Jul 2006 21:53:02 GMT
[View Forum Message](#) <> [Reply to Message](#)

unodgs wrote on Sat, 08 July 2006 22:32Werner wrote on Sat, 08 July 2006 16:08
If the above code doesn't help (it doesn't comply with the C++ standard)

May I know what's wrong with it??

Sure! You have every right to know!

Here we go:

1.
dr_jumba wrote that he needs to pass the command line arguments to a 3rd party library. It seems much more likely that the "3rd party" expects these arguments to be passed in the standard C++ way than in the Ultimate++ way.

2.
So lets compare the results of your code with what the C++ standard expects:

a)
The first argument (argv[0]) expected by the C++ standard is the (path and) name of the running application. If this is not available the standard expects "argv[0][0] == '\0'".

Your code passes the 1st "real" argument as `argv[0]`. It suggests itself that this might mislead the receiving library.

b)

The standard expects `"argv[argc] == (char* 0)"`.

I'm not sure whether your code guarantees that and this missing final pointer could terribly crash the library and everything else.

c)

`argv` itself and the strings to which it points must be modifiable.

You use `"const char **argv"`.

d)

Finally the "application name issue" makes your `cnt (= argc)` carry a wrong value - 1 too few.

Apologies for this lecture!

Werner
