Subject: Re: .ini file helpers
Posted by dolik.rce on Tue, 10 Jul 2012 19:14:42 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Today I tried to use the Ini* helper functions, but I found out that they are not very intuitive. I believe that with a little work, they might become much more useful.

I see these problems:
1) INI_DOUBLE is missing.

2) Once you call GetIniKey, the file is loaded and it is not possible to reload configuration. In a daemons/services, it is often desirable to reload config without restarting.

3) There are currently two separate mechanisms SetIniFile+GetIniKey function and the Ini namespace with INI_* macros. I didn't notice any connection between those two, but they are both shown in the reference/INI example.

Now tell me where I'm wrong If I'm not, then I think all these problems are solvable.

The first one is trivial.

The second could be solved simply by using global static VectorMap for the key-value pairs and adding ReloadIniFile() function. Something like:static StaticMutex sMtx;

```
static char  sIniFile[256];
static bool s_ini_loaded;
static VectorMap<String, String> s_ini_key;

void SetIniFile(const char *name) {
 Mutex::Lock __(sMtx);
 strcpy(sIniFile, name);
}

void ReloadIniFile(const char *name) {
 Mutex::Lock __(sMtx);
 if(*name)
  strcpy(sIniFile, name);
 s_ini_loaded = true;
 s_ini_key = LoadIniFile(*sIniFile ? sIniFile : ~ConfigFile("q.ini"));
 #ifdef PLATFORM_WIN32
  if(s_ini_key.GetCount() == 0)
   s_ini_key = LoadIniFile(~GetExeDirFile("q.ini"));
  if(s_ini_key.GetCount() == 0)
   s_ini_key = LoadIniFile("c:\\q.ini");
 #endif
 #ifdef PLATFORM_POSIX
  if(s_ini_key.GetCount() == 0)
```

```
   s_ini_key = LoadIniFile(GetHomeDirFile("q.ini"));
 #endif
}

String GetIniKey(const char *id, const String& def) {
 if(!s_ini_loaded)
  ReloadIniFile(sIniFile);
 return s_ini_key.Get(id, def);
}
```

But I think there is better solution and it is related to the problem 3). I propose to drop the GetIniKey function altogether (or keep it just as wrapper for backward compatibility). If we rewrite LoadIniStream to guess value types (not very difficult) and store them as Value instead of String AND add a reference to original Ini::IniXYZ variable into the IniInfo struct, we could iterate through IniInfos after each reload and update the values of the variables in the Ini namespace. In this way, the values declared in code would be possible to overwrite by values from INI file, even at multiple times if desired. Also, GetIniKey would be unnecessary, as Ini::MyVar would be always up to date and shorter to write.

I'm not sure if I expressed myself clear enough, ask if there is anything non-understandable or if you see any unsolvable problem within this proposal...

Best regards,
Honza