Subject: Re: Some Experiment with Size of Upp Executable
Posted by Lance on Mon, 27 Dec 2021 15:51:09 GMT
View Forum Message <> Reply to Message

Hi Novo:

Thank you for the info. Could you give a quick link to introduce me to map-file tools, etc?

I was trying to figure out how parts of Core are pulled into the final executable to make what it is.

Turns out plugin/z, the only plugin Core used, should not be blamed.

In <plugin/z/lib/crc32.h>, I manage to encapsulate
local const z_crc_t FAR _crc_table[TBLS][256];
in a function call; it will increase the executable size of Blank Core project to 769024 b (I used only MSBT22x64, as it outperforms CLANG) from 767488. inline the function doesn't change anything: so there is indeed potential cost on size by hiding it in a function if it's used. Then I removed reference to it by changing relevant lines in <Core/App.h> to something like

```
#define CONSOLE_APP_MAIN \
void ConsoleMainFn_(); \
 \
int main(int argc, char *argv[]) { \
/* UPP::AppInit__(argc, (const char **)argv); \
 UPP::AppExecute__(ConsoleMainFn_); \
 UPP::AppExit__(); \
 return UPP::GetExitCode(); */\
} \
 \
void ConsoleMainFn_()
```

to produce a do-nothing main(), now the executable size go down to 745472 bytes. So the cost is definity in Core itself.

As a contrast, a blank C++(no U++) console produces an executable of 109056B.

These are cost paid for U++ facilities, and its 0 or all, not pay as you go.

I still do not have a answer to my question, but it seems not worth pursuing any further.