Subject: [BUG] TheIDE prevents using MSC Incremental link
Posted by Shire on Sun, 24 Feb 2013 11:59:34 GMT
View Forum Message <> Reply to Message

Incremental linking is very good think to accelerate debug builds and enabled by default in Microsoft compilers.
But TheIDE prevents using this technology.

LINK : app.exe not found or not built by the last incremental link; performing full link
Why?
Quote:LINK performs a full link if any of the following situations occur:

The incremental status (.ilk) file is missing. (LINK creates a new .ilk file in preparation for subsequent incremental linking.)
There is no write permission for the .ilk file. (LINK ignores the .ilk file and links nonincrementally.)
The .exe or .dll output file is missing.
The timestamp of the .ilk, .exe, or .dll is changed.
A LINK option is changed. Most LINK options, when changed between builds, cause a full link.
An object (.obj) file is added or omitted.
An object that was compiled with the /Yu /Z7 option is changed.
uppsrc/ide/Builders/Build.cpp
bool MakeBuild::Build
<...>
// Set the time of target to start-time, so that if any file changes during
// compilation, it is recompiled during next build
SetFileTime(target, start_time);
Changing target file timestamp is bad practice. Changing sources during compilation too . Hope you avoid this.

Also there is fully useless invoking of manifest tool:
app.exe.manifest : general error c1010070: Failed to load and parse the manifest.
uppsrc/ide/Builders/MscBuilder.icpp
bool MscBuilder::BuildPackage
<...>
```
if((IsMsc86() || IsMsc64()) && is_shared) {
  String mt("mt -nologo -manifest ");
  mt << GetHostPathQ(product + ".manifest") << " -outputresource:" << GetHostPathQ(product)
<< ";2";
    Execute(mt);
}

bool MscBuilder::Link
<...>
if((IsMsc86() || IsMsc64()) && HasFlag("SO")) {
  String mt("mt -nologo -manifest ");
   mt << GetHostPathQ(target + ".manifest") << " -outputresource:" << GetHostPathQ(target)
      << (HasFlag("DLL") ? ";2" : ";1");
   Execute(mt);
```

}
You can avoid error message by putting manually manifest file near resulting binary. But modern linker automatically (and by default) generates it for resulting executable.
If you want embed custom manifest, you can do it by linker options (/MANIFEST) in Package organizer.
I put these small modifications with proper support of precompiled headers here.
