Subject: Automated tests
Posted by ebojd on Thu, 01 Feb 2007 20:16:12 GMT
View Forum Message <> Reply to Message

If anyone has suggestions, and/or examples, for building attached automated test suites please post (I am in the early stages of a large project, and have started a test suite for the primitives, and would like to integrate it with the GUI app).

Subject: Re: Automated tests
Posted by mirek on Mon, 05 Feb 2007 09:59:25 GMT
View Forum Message <> Reply to Message

ebojd wrote on Thu, 01 February 2007 15:16If anyone has suggestions, and/or examples, for building attached automated test suites please post (I am in the early stages of a large project, and have started a test suite for the primitives, and would like to integrate it with the GUI app).

Well, this is really issue that is being considered and discussed here for some time.

After long thinking about the issue, I plan to implement following simple framework:

Tests will be placed in separate nest (e.g. 'upptests'). Each package will represent single test.

TheIDE will get a new function that will compile and run all tests, most likely with ".TESTING" added to compilation flags. This flag will change behaviour of ASSERT (and likes) to simply abort the test and return non-zero exit code. Zero exit code will be considered "test passed".

Later we might add some more automatization. I think the final stage should be configurable to run on server, sync repository during the night, do tests and email results to developers.

It is BTW possible to do this now as TheIDE has commandline mode. In development we are using "BuildAll" package that builds all examples in Win32; it is simple single C++ file:

```
#include <Core/Core.h>

using namespace Upp;

// This is diagnostic package:
// It compiles all U++ examples using MSC8 and MINGW build methods
// or methods listed on commandline

String input = "d:\\";
String output = "e:\\all";
```

```
String umk = "d:\\theide\\umk.exe ";
Vector<String> bm;

Vector<String> failed;

void Build(const char *nest, const char *bm, bool release)
{
 String flags = release ? "r" : "b";
 String mn = release ? "R" : "D";
 String n = String().Cat() << nest << '-' << bm << '-' << mn;
 Cout() << n << '\n';
 String outdir = AppendFileName(output, n);
 DeleteFolderDeep(outdir);
 RealizeDirectory(outdir);
 FindFile ff(AppendFileName(AppendFileName(input, nest), "*.*"));
 bool first = true;
 while(ff) {
  if(ff.IsFolder()) {
   String txt;
   txt << ff.GetName() << ' ' << bm << ' ' << mn;
   Cout() << "  Building " << txt;
   String c;
   c << umk << nest << ' ' << ff.GetName() << ' ' << bm << " -I" << flags;
   if(first)
    c << 'a';
   c << ' ' << outdir;
   if(system(c)) {
    Cout() << " *** FAILED *** !\n";
    failed.Add() << txt;
   }
   else {
    Cout() << " ok\n";
    first = false;
   }
  }
  DeleteFile(AppendFileName(outdir, ff.GetName() + ".ilk"));
  DeleteFile(AppendFileName(outdir, ff.GetName() + ".pdb"));
  ff.Next();
 }
}

void Build(const char *nest, bool release)
{
 for(int i = 0; i < bm.GetCount(); i++)
  Build(nest, bm[i], release);
}

void Build(const char *nest)
```

```
{
 Build(nest, false);
// Build(nest, true);
}

CONSOLE_APP_MAIN
{
 const Vector<String>& arg = CommandLine();
 input = GetDataFile("BuildAll.cpp")[0] + String(":\\");
 output = "C:\\out";
 for(int i = 0; i < arg.GetCount(); i++)
  bm.Add(arg[i]);
 if(bm.GetCount() == 0) {
  bm.Add("MSC71cdb");
//  bm.Add("MSC8cdb");
//  bm.Add("MINGWI");
 }
 Build("examples");
 Build("reference");
 Build("tutorial");
 if(failed.GetCount()) {
  Cout() << "***** Failed builds: \n";
  for(int i = 0; i < failed.GetCount(); i++)
   Cout() << "  " << failed[i] << '\n';
 }
 RDUMPC(failed);
}
```

Subject: Re: Automated tests
Posted by ebojd on Mon, 05 Feb 2007 16:01:21 GMT

View Forum Message <> Reply to Message

Thanks luzr,

I'll look into this. It sounds similar to what I typically write into my test suites. As a note, when testing I typically do not halt on error, but report the test and error and continue on -- so if there are more than one error I get an end report of all errors. This should be easily configurable (similar to the current build system).

For organization I typically put my own test programs in a "test" subdirectory of the specific package/module/library to make sure they are physically together. This probably makes no sense with U++, but if someone gives a copy of the program away I want to make sure they have the associated test suite (for that specific version of the source). This also allows me the run "make check" which recursively traverses the source tree and runs all tests.

Is there any documentation available for your testing framework?

EBo --

Subject: Re: Automated tests
Posted by mirek on Mon, 05 Feb 2007 17:42:02 GMT
View Forum Message <> Reply to Message

ebojd wrote on Mon, 05 February 2007 11:01
As a note, when testing I typically do not halt on error, but report the test and error and continue on -- so if there are more than one error I get an end report of all errors. This should be easily configurable (similar to the current build system).

Yes, but note that system of packages makes nice alternative. Packages can be really small (e.g. I have 315 packages in my uppdev nest and the whole mess is still manageable).

Quote:
Is there any documentation available for your testing framework?

There is no testing framework yet, just the basic idea which I have explained in my post This is "Technology lab"...

Mirek
