Arcabouço de Apoio ao Teste Automatizado para Programas Redigidos em C
Leia.me

Arndt von Staa

Março 2006

Apresentação

Este texto descreve, resumidamente, o conteúdo do arquivo `ArcaboucoTeste-2-00.zip` que contém toda a documentação e todos os módulos fonte, exemplos e utilitários utilizados pelo *Arcabouço de Apoio à Automação dos Testes de Programas Redigidos em C*: `AutoTest`.

Instalação

Crie um diretório no qual você deseja instalar o arcabouço. *Dezipe* o arquivo `ArcaboucoTeste-2-00.zip` assegurando que sejam gerados os sub-diretórios contidos no arquivo `.zip`. Isto criará toda a estrutura de diretórios utilizada e copiará tanto o arcabouço como os exemplos de uso. São copiados também os *scripts* de construção (`make`) e teste do próprio arcabouço.

Estrutura de diretórios

Ao *dezipar* são gerados os seguintes diretórios:

`Simples` contém um exemplo de uso simples. Este exemplo não faz uso da instrumentação disponibilizada pelo arcabouço.

`Documentos` contém todos os documentos que acompanham o arcabouço de teste.

`Ferramnt` contém todas as ferramentas (programas) complementares. Entre eles encontra-se `GMake`, um programa para gerar scripts de `make` a partir de arquivos que descrevem a composição de construtos.

`Instrum` contém um exemplo de uso do arcabouço em que se utilizam as ferramentas de instrumentação (controle de completeza e controle de acesso a espaços de dados dinâmicos) disponíveis.

`Arcabouc` contém todos os arquivos necessários para testar o arcabouço e criar a biblioteca.

`Tabela` contém um exemplo de criação e acesso a tabelas de *strings*. Estas tabelas permitem concentrar em um único local todos os *strings* utilizados em um programa, facilitando, assim a sua conversão para outros idiomas. Este exemplo ilustra como criar scripts de `make` envolvendo uma variedade de ferramentas desenvolvidas pelo usuário. Ilustra ainda o uso de tabelas de implementação, tabelas de definição e tabelas de especificação.

Criação da biblioteca

A biblioteca do arcabouço precisa ser compatível com a versão da plataforma de desenvolvimento utilizada. Portanto, antes de utilizá-la, ela precisa ser recompilada[1]. Para tal proceda da seguinte forma:

- no diretório `.\arcabouc\batches` execute o batch `compilatudo.bat`, isto recompila todos os construtos de teste do arcabouço inclusive a biblioteca. Este passo somente é necessário se forem testados os construtos de desenvolvimento do arcabouço.
- nos diretórios `.\simples`, `.\tabela` ou `.\arcabouc\instrum` execute o batch `crialib.bat` e a seguir o batch `compilatudo.bat`, isto recompila todos os construtos de cada um dos 3 exemplos usando a biblioteca nova. Evidentemente só precisa ser realizado nos exemplos que serão examinados.

Para que a compilação se dê corretamente em uma janela de linha de comando do Windows (`CMD`), é necessário que o contexto (*environment*) esteja corretamente inicializado para a plataforma em uso. Caso não compile (ex.: não encontra o programa `nmake`) procure o *batch* `vsvars32.bat` no diretório `common\tools` da plataforma de compilação a ser utilizada (`"visual studio"\vc`) e o execute. Note que isto terá que ser realizado sempre que uma nova janela de linha de comando for aberta. Por esta razão é conveniente ter uma versão local deste *batch*.

Para que a compilação se dê corretamente no IDE, são necessários diversos ajustes nas opções do projeto. Em especial, é fundamental tornar OFF a chave "pré-compiled headers" do compilador. Ao invés de usar a biblioteca, pode-se incorporar os fontes da biblioteca ao projeto, junto com os módulos do construto sendo elaborado.

Documentação

- `ArcaboucoTeste-2-00.pdf` – contém a documentação do arcabouço de apoio ao teste automatizado de programas redigidos em C. Cabe salientar que a documentação detalhada se encontra nos diversos módulos de definição.
- `ArcaboucoTeste-2-00-LeiaMe.pdf` – é a versão eletrônica do presente documento.
- `GMAKE-v6-00.pdf` – é a documentação da versão 6.00 do programa `GMake`. Este programa é utilizado para, a partir de um arquivo que descreve a composição de um construto, gerar as diretivas `.make` e `.build` usadas para compilar este construto. O `GMAKE` permite criar scripts de `make` que envolvem uma variedade de ferramentas criadas pelo usuário, além das ferramentas que compõem o compilador.

Arquivos de código fonte que compõem o arcabouço

Os arquivos descritos nessa seção, em particular os módulos de definição (`.h`) complementam e detalham a documentação. Esse conjunto de módulos compõe a biblioteca estática `ArcaboucoTeste.lib`.

---

[1] A versão do Visual Studio que eu estou usando é a 7. As que se encontram no LabGrad são 6 e 8. Em ambos os casos existem incompatibilidade entre as versões dos módulos objetos gerados pelos compiladores.

- PRINCIP.C – é o módulo de implementação do programa principal. O presente arcabouço está projetado para sempre prover o programa principal. Caso isto não seja desejado, pode-se reconstruir a biblioteca sem incluir este módulo. Neste caso a ativação do arcabouço se dá através da função TST_ControlarSessaoTeste contida no módulo GENERICO.

- GENERICO.C, GENERICO.H – são os módulos de implementação e de definição do módulo de teste genérico do arcabouço.

- LERPARM.C, LERPARM.H – são os módulos de implementação e de definição do módulo de leitura e análise de comandos de teste. Todos os comandos de teste devem ser lidos com este módulo. Ele provê função de análise dos comandos de teste e permite a mistura de parâmetros literais e simbólicos (nomes declarados), tornando, assim, mais legíveis os scripts de teste. GENERICO e LERPARM formam um par de módulos fortemente interdependentes. A razão da separação é assegurar que cada qual trate de um assunto (*concern*) bem definido.

- TST_ESPC.H – é o módulo de definição de todos os módulos de teste específicos que porventura venham a ser desenvolvidos. O uso deste módulo de definição assegura a correta interface entre o módulo GENERICO e o módulo de teste específico que, na realidade, estende (herança) aquele módulo.

- CESPDIN.C, CESPDIN.H - são os módulos de implementação e de definição do módulo de controle de acesso a espaços dinâmicos. Inclua CESPDIN.H nos módulos a testar em que você deseja controlar o acesso, sempre controlado por #ifdef _DEBUG. Caso o módulo a testar contenha chamadas para funções desse módulo, estas também deverão estar controladas por _DEBUG.

- CONTA.C, CONTA.H - são os módulos de implementação e de definição do módulo de controle da completeza dos testes. O comando de inclusão de CONTA.H bem como as chamadas a funções desse módulo (por exemplo, CNT_CONTA) contidas no módulo a testar, devem sempre estar contidos em um controle #ifdef _DEBUG.

- GERAALT.C, GERAALT.H – são os módulos de implementação e de definição do módulo que fornece números aleatórios. Podem ser gerados números aleatórios de uma distribuição uniforme, bem como de uma distribuição em que estão definidos intervalos de freqüência. É freqüente que se deseje gerar testes automaticamente ao invés de criá-los manualmente. Nestes casos é interessante gerá-los de forma aleatória. Este módulo foi projetado de modo que possa ser utilizado independentemente do arcabouço de teste. Portanto, pode continuar a ser utilizado na versão de produção.

- INTRPCED.C, INTRPCED.H - são os módulos de implementação e de definição do módulo de interpretação de comandos de teste voltados ao controle de acesso a espaços dinâmicos. Esse módulo interpreta somente os comandos que podem ser utilizados sem um contexto específico de execução do módulo CESPDIN. Os outros comandos de teste deste módulo encontram-se no módulo TESTCED. Módulos de teste específico, bem como os módulos a testar, podem utilizar as funções disponibilizadas pelo módulo CESPDIN. No módulo a testar tais chamadas devem estar em um controle de compilação condicional, de modo que não façam parte do módulo compilado para produção.

- INTRPCNT.C, INTRPCNT.H - são os módulos de implementação e de definição do módulo de interpretação de comandos de teste voltados ao controle da completeza do teste. Esse módulo interpreta somente os comandos que podem ser utilizados sem um contexto específico de execução do módulo CONTA. Os outros comandos são interpretados pelo módulo de teste específico TESTCNT.

- TABSIMB.C, TABSIMB.H – são os módulos de implementação e de definição do módulo que manipula as tabelas de símbolos requeridas por outros módulos do arcabouço. Este módulo foi projetado de modo que possa ser utilizado independentemente do arcabouço de teste. Portanto, pode continuar a ser utilizado na versão de produção.
- TESTGEN.C – é o módulo de implementação do módulo de teste específico do módulo GENERICO. Este módulo testa as funções de comparação contidas no módulo GENERICO.
- TESTCED.C – é o módulo de teste específico do módulo CESPDIN.
- TESTCNT C – é o módulo de teste específico do módulo CONTA.
- TESTLER C – é o módulo de teste específico do módulo LERPARM.
- TESTTBS C – é o módulo de teste específico do módulo TABSIMB.
- IdTiposEspaco.def – (está no diretório: .\Arcabouc\Tabelas) é um arquivos de declaração (enum) dos identificadores de tipos de espaços dinâmicos de dados. Cada módulo deve definir o seu conjunto de identificadores. Através da inclusão dos nomes desses identificadores no arquivo IdTiposEspacos.def, cria-se um conjunto de identificadores únicos, sem requerer a recompilação dos módulos do arcabouço.

Ao desenvolver um módulo de teste específico, devem sempre ser incluídos os módulos de definição GENERICO.H, LERPARM.H e TST_ESPC.H. Já ao desenvolver um módulo da aplicação (os módulos a testar) podem ser incluídos GERAALT.H e TABSIMB.H. Podem ser incluídos também CESPDIN.H e/ou CONTA.H, no entanto, estes deverão sempre estar envolvidos em um controle de compilação condicional (#ifdef ... #endif), como também devem estar as chamadas a funções por eles disponibilizadas.

Arquivos de código fonte que compõem o exemplo

Estes módulos são substituídos pelos módulos a testar e correspondentes módulos de controle de teste específicos.

- ARVORE.C, ARVORE.H – são os módulos de implementação e de definição do exemplo, um módulo de edição e acesso a árvores binárias.
- TESTARV.C – é o módulo de implementação do módulo de teste específico do módulo editor de árvore ARVORE. São fornecidas duas versões. Uma simples que não faz uso da instrumentação e trata de uma única árvore, encontra-se no diretório .\simples. A segunda trata de várias árvores simultaneamente e faz uso de instrumentação. Encontra-se no diretório .\instrum\fontes.

Arquivos de diretivas para a compilação em .\Simples

- ExemploSimples.comp – é o arquivo de diretivas de composição do programa exemplo. Esse arquivo é utilizado por GMAKE para gerar as diretivas de MAKE. Este arquivo de diretivas destina-se a gerar a versão de produção. São gerados os arquivos:
  - ExemploSimples.MAKE – é o arquivo de diretivas MAKE utilizado para compilar a versão de produção do exemplo.

- ExemploSimples.BUILD – é o arquivo de diretivas para ligar o programa exemplo, versão produção, a partir de seus módulos.
- ExemploSimples.EXE – é o programa compilado, versão produção, do exemplo. Este arquivo é gerado pelo comando de linha Compila TesteArvore. Ao mudar da versão de depuração para a versão de produção ou vice-versa, destrua primeiro todos os módulos objeto (.obj ou .o) pois, de outra forma, alguns, ou até todos, poderiam não ser recompilados, gerando confusão na hora de ligar ou testar.
- ExemploSimples.LIST – é a listagem da composição do programa exemplo.

- ExemploSimplesDBG.comp – é o arquivo de diretivas de composição do programa exemplo, versão depuração. Esse arquivo é utilizado por GMAKE para gerar as diretivas de MAKE. São gerados os arquivos:
  - ExemploSimplesDBG.MAKE – é o arquivo de diretivas MAKE utilizado para compilar a versão de depuração do exemplo. Os módulos a serem compilados para depuração estão marcados com /D_DEBUG no arquivo de diretivas de composição ExemploSimplesDBG.comp.
  - ExemploSimplesDBG.BUILD – é o arquivo de diretivas para ligar o programa exemplo, versão depuração, a partir de seus módulos.
  - ExemploSimplesDBG.EXE – é o programa , versão depuração, do exemplo. Este arquivo é gerado pelo comando de linha Compila ExemploSimplesDBG.
  - ExemploSimplesDBG.LIST – é a listagem da composição do programa exemplo.

Arquivos de composição do arcabouço em .\Arcabouc\Composicao

O arcabouço de teste foi desenvolvido e testado com ele mesmo. Durante o desenvolvimento foram criados módulos de enchimento (*stubs*) posteriormente substituídos por módulos de produção completos. À medida que esta substituição ia sendo realizada, os scripts de teste foram desenvolvidos ou ampliados. O diretório .\Arcabouc reflete o resultado final deste processo. Os *scripts* de composição dos construtos, de make e de teste são fornecidos, pois estabelecem o ambiente de desenvolvimento necessário para se poder evoluir o arcabouço com testes minimamente bem definidos (cobertura de chamada de função assegurada por ferramentas de geração de código). Os nomes dos arquivos de diretivas de composição são suficientes para saber a finalidade dos construtos gerados. A mesma observação vale para os nomes dos *scripts* de teste.

Arquivos de diretivas de teste

- *.script – são os arquivos de diretivas de teste (*script*) que ilustram o uso do presente arcabouço. Diversos desses arquivos de diretivas têm por objetivo testar módulos que compõem o próprio arcabouço. Do ponto de vista da qualidade dos teste estes exemplos são superficiais. Dependendo da plataforma utilizada, o *script* de teste TesteControleEspaco-Voa.script pode cancelar a execução do programa, uma vez que deturpa ponteiros
- *.count – são os arquivos contendo os identificadores de contadores utilizados ao testar o módulo contador e ao testar o módulo árvore.

- `*.estat` – são arquivos contendo estatísticas acumuladas gerados ao realizar vários testes em uma única execução (`testatudo.bat`)
- `*.log` – são os arquivos gerados ao realizar um teste. Eles contém o laudo do teste.

Utilitários e ferramentas

- `GMAKE.EXE` – é um programa utilitário que gera o arquivo de diretivas `MAKE` a partir de uma descrição de composição de um construto. A sua documentação está em `GMAKE-v6-00.pdf`.
- `GMake.parm` – é o arquivo de diretivas para o `GMake` utilizado para gerar diretivas `MAKE` para o projeto arcabouço de teste utilizando a plataforma MS-Visual C/C++. Este arquivo deve ser ajustado para poder compilar visando outras plataformas e outros projetos. Consulte a documentação de `GMake` incluída neste pacote.
- `GeraMake.bat` – é um *batch* utilizado para gerar as diretivas `MAKE` de um determinado construto exemplo.
- `CriaLib.bat` – gera a biblioteca a partir dos fontes contidos no diretório `.\arcabouc\fontes` e a copia para o diretório do exemplo em questão.
- `GeraTudo.bat` – gera as diretivas `MAKE` de todos os construtos exemplo.
- `Compila.bat` – compila um programa utilizando um arquivo de diretivas de `MAKE` geradas por `GMAKE`. Este batch assume que o ambiente (*environment*, `set`, `path`) esteja corretamente configurado para que o compilador a ser utilizado compile corretamente. Caso esteja utilizando MS Visual Studio 6 ou mais recente, procure o batch `vsvars32.bat` e verifique se o *environment* reflete as declarações nele contidas.
- `CompilaTudo.bat` – compila todos os construtos exemplo utilizados para testar o arcabouço e o módulo exemplo árvore.
- `Testa.bat` – testa um determinado construto com um *script* específico.
- `TestaTudo.bat` – realiza todos os testes versão de produção. Para que este teste termine com 0 falhas é necessário que `ARVORE` e `TESTESPC` tenham sido compilados com o parâmetro `/_DEBUG` **não** definido. Caso os módulos objeto tenham sido gerados usando `TestaTudoDebug`, ou tenham sido compilados com a chave `/D_DEBUG` ligada, é necessário excluí-los e recompilar tudo. Razão: o `MAKE` não controla *como* foram compilados os módulos, controlando somente *quando* o foram.
- `LimpaTudo.bat` – elimina todos os arquivos que são gerados por intermédio de alguma ferramenta.
- `Tudo.bat` – limpa tudo, gera todos os *scripts* de `MAKE`, compila todos os construtos e realiza todos os testes.
- `EXBESTATC.C`, `EXBESTAT.EXE` – é um pequeno programa que permite exibir o arquivo de acumulação das estatísticas de teste. Quando um teste for formado por diversas massas de teste (*scripts* de teste) torna-se conveniente acumular as estatísticas relativas a cada massa, de modo que, ao final do teste de todas elas, possa-se exibir o resultado acumulado destas estatísticas. O arquivo `TestaTudo.bat` ilustra o uso desse programa.

- `CompileBanner.C`, `CompileBanner.exe` – é um pequeno programa que exibe um separador no vídeo. O programa é utilizado para exibir separadores ao compilar construtos, facilitando a compreensão de uma compilação agregada de diversos construtos.

FIM