Engenharia de Software : Ferramentas para Empacotamento

This page last changed on Apr 21, 2009 by amadeu.

Aqui explicamos rapidamente a principais funcionalidades das ferramentas de empacotamento esclarecendo quais requisitos elas cumprem. Posteriormente, é recomendado a leitura dos Procedimentos de Empacotamento para entender como usar e configurar todos os assistentes.

O que fazem os assistentes de empacotamento?

Organização de diretórios padrões

Adota-se uma estrutura padrão de diretórios ajuda na uniformização das configurações de ambiente (variáveis shell, localização, nomes de pastas, etc) o que facilita a troca de informações entre os desenvolvedores, diminui as chances de erros nas etapas de configuração do ambiente e facilita o aprendizado para novos integrantes.

Como há diferentes atividades a serem desempenhadas durante um ciclo de desenvolvimento, as ferramenta consideram três **cenários básicos** (motivam a existência de diretórios em um nível mais alto daquela hierarquia):

- **desenvolvimento**: locais onde manter os códigos-fonte do barramento e das bibliotecas externas;
- **compilação e testes frequentes**: local onde manter os binários gerados pela compilação e de onde devem ser executados os testes mais frequentes no dia-a-dia dos desenvolvedores;
- **validação**: local onde testar a instalação dos pacotes gerados antes do release para o cliente.

Automatização das atividades básicas na implantação

Os assistentes destinam-se a automatizar tarefas antes feitas manualmente durante um ciclo de desenvolvimento, a exemplo de:

1. definir quais configurações de ambiente (shell, diretórios, etc) são necessárias para compilar, gerar pacotes e executar o barramento
2. definir quais flags de compilação para cada plataforma
3. (re-)compilar tanto as bibliotecas externas quanto os códigos funcionais do barramento
4. gerar pacotes do OpenBus e suas dependências para serem entregues aos clientes
5. instalar os pacotes do OpenBus
6. testar e executar o OpenBus

Desta forma, existem três assistentes complementares:

- compilação : responsável pela compilação dos códigos encapsulando os comandos específicos para compilar cada software de acordo com seu tipo de ferramenta de *building* (autotools, tecmake ou outra)
- geração de pacotes : útil para gerar pacotes versionados contendo 1 ou mais softwares básicos (bibliotecas, serviços básicos ou outros)
- instalação : simplifica a instalação e realiza verificações em tempo de instalação como a compatibilidade binária do pacote sendo instalado

O uso desses assistentes é melhor detalhado no manual dos Procedimentos de Empacotamento.

Suporte a multi-plataforma

Além da linguagem Lua ajudar na portabilidade, principalmente na manipulação de arquivos, uso de pipes e casamento de padrões das saídas em tela, os assistentes possuem uma estratégia própria de portabilidade.

A maior parte dos comandos de sistema executados pelos assistentes estão portados para diversas plataformas como Linux, Solaris, IRIX e MacOSX de forma produzir um único *output* para os códigos que o executam. As tabelas que mapeiam esses comandos em utilitários do sistema encontram-se no arquivo **trunk/tools/lua/tools/platforms.lua** podendo ser facilmente estendidas.

Suporte a descrição de pacotes multi-plataforma

Entende-se por **descrição de pacotes** um conjunto de tabelas Lua que revelam detalhes importantes para automatizar as tarefas de compilação, geração e instalação. Os descritores de pacotes devem ser versionados junto ao código principal do OpenBus.

Veja também o detalhamento completo dos campos na descrição de um pacote.

Observe que uma descrição de pacote contiver os campos **dev_files** e **conf_files**, isto implica necessariamente a existência de pacotes com sufixos **-dev** e **-conf**, respectivamente. Essa abordagem permite a separação, por exemplo, do quê distribuir como binário do pacote (binário do openssl) ou como artefatos para desenvolvimento (libs estáticas e arquivos de cabeçalhos do openssl).

Suporte a perfis diferenciados de pacotes

Os **perfis de pacotes** são arquivos onde indica-se quais pacotes formam um **perfil** de empacotamento que ajuda a organizar a constituição de um *release* do OpenBus e seus softwares dependentes. Esse conceito é similar aos meta-pacotes em outros sistemas de pacotes (como em .deb e .rpm).

Um exemplo de um perfil de pacote está localizado no subversion no arquivo **trunk/tools/profiles/admin** e indica um perfil **admin** destinado a administradores do barramento. Seu conteúdo referencia o nome dos pacotes: **openssl-0.9.9**, **lua5.1**, **lposix**, **luasocket**, **oilall**, **luuid**, **lualdap**, **scsall**, **lce**, **openbus-core** e **openbus-core-conf**.

É importante observar que alguns softwares descritos produzem mais de um pacote como a descrição do **openbus-core** que produz os pacotes **openbus-core-dev** e **openbus-core-conf**.

Assim é possível dispor de outros perfis para usos específicos, por exemplo, o perfil **develorbix** só é constituído pelos pacotes **scsorbix**, **openbus-orbix-dev**, **openbus-orbix-test** e **openbus-demo-orbix-dev**.

Suporte a diferentes ferramentas de compilação

O campo **type** da tabela **build** na descrição do pacote permite a seleção de tratadores diferenciados para cada *backend* de compilação. Atualmente existe suporte a 2 *building backends*: **tecmake** e **autotools**. Cada um desses tipos podem especificar campos diferenciados na tabela de **build**.

Veja também o detalhamento completo dos campos na descrição de um pacote.

Suporte a versões nos pacotes gerados

O assistente de geração de pacote cria arquivos comprimidos cujo nome adota a seguinte especificação: **openbus-PERFIL-VERSAO-PLATAFORMA.tar.gz**

- **PERFIL** : nome do perfil de pacote, exemplo: admin
- **VERSAO** : identificador da versão, exemplos: OB_HEAD_r9999, OB_v1_03_2009_03_10
- **PLATAFORMA** : identificador da plataforma conforme a notação Tecmake (seção variáveis pré-definidas), exemplos: Linux26g4, Linux24g3, SunOS58

Particularmente, o identificador da versão é formado por:

- **OB** — acrônimo para *O*pen*B*us
- **HEAD_r9999** — onde HEAD significa que o pacote foi gerado a partir do **trunk** do subversion cuja última mudança ocorreu na revisão **9999**
- **v1_03_2009_03_10** — que significa que o pacote foi gerado a partir de um **branch** ou **tag** específicos

Dessa forma, para saber todas as versões existentes do OpenBus é preciso olhar os branches e tags existentes.

Veja outras informações no manual do assistente de geração de pacotes. Vale ressaltar que é necessário reconfigurar os assistentes para considerar a compilação e geração em uma determinada versão.

Suporte a configuração dos pacotes em tempo de instalação

Os assistentes permitem a criação de modelos (*templates*) de configurações e ações personalizadas. Esse modelos serão usados pelo instalador para **solicitar dados** ao usuário no momento pré-instalação. Esses dados podem ser tratados por **ações customizadas** (*hooks*) no momento pós-instalação.

Para usar esse recurso, cada descrição de pacote pode preencher os campos **conf_template** e **conf_files**. O primeiro indica o arquivo de modelo que será **automaticamente** incluído nos metadados dos pacotes gerados para que o instalador proceda às etapas de pré e pós instalação adequadamente. O segundo indica os arquivos de configuração que o pacote carrega para permitir um tratamento adequado durante, por exemplo, uma futura re-instalação.

Para definir **ações customizadas** basta definir uma função Lua com a seguinte assinatura:

```
function configure_action (answers, tempdir, util)
-- answers = contém a lista de propriedades preenchidas pelo usuário
-- tempdir = contém o caminho do diretório temporário usado pela instalação
-- util = referência para o objeto com os métodos utilitários
end
```

Suporte a repositórios de pacotes

Os pacotes com os fontes das bibliotecas externas podem ser armazenados em repositórios remotos. O campo **source** das descrições dos pacotes permitem o preenchimento da URL de origem do pacote. É possível estender o suporte a diferentes protocolos de acesso aos repositórios. Por enquanto, só existe suporte para **http**, mas é interessante implementar suporte a outros protocolos (svn, cvs, ssh) como faz o LuaRocks.

O suporte ao protocolo **http** permite indicar arquivos comprimidos nos formatos **.tar.gz**, **.tgz**, **.tar.bz2**, **.tbz2** e **.zip**.

Após a obtenção, esses arquivos são mantidos no diretório **openbus/packs** e são extraídos no diretório **openbus/lib**. Esses diretórios podem ser alterados de acordo com as variáveis globais de configuração dos assistentes.

Veja mais detalhes do repositório e da forma de baixar os pacotes durante a compilação.

Suporte a evolução das descrições dos pacotes e das configurações

Conforme as funcionalidades listadas acima, é possível identificar que os assistentes para empacotamento podem ser configurados em vários níveis. Dessa forma, há diversos arquivos de entrada para os assistentes que devem ser versionados em conjunto com o projeto em questão.

A informação sobre **quais são os pacotes** e **como empacotá-los** são mantidas nos arquivos de descrição (**.desc**) e nos perfis de pacotes (diretório **trunk/tools/profiles**). Os **modelos de configuração** (diretório **trunk/tools/templates**) também estão intimamente relacionados a um subconjunto de pacotes. Por fim, **configurações do console** (diretório **trunk/tools/shell**) e **configurações dos assistentes** completam a lista de conteúdos que devem ser versionados por projeto.

Desta forma, o código funcional dos assistentes se restringe aos diretórios **trunk/tools/lua** e **trunk/tools/src** que não necessariamente precisam estar versionados em conjunto com o projeto do OpenBus.

Suporte a desinstalação e re-instalação

O assistente de compilação gera metadados para cada pacote sendo compilado. Esses metadados indicam quais arquivos compõem cada pacote ou suas variações (**-dev**, **-conf** e **templates**).

Desta forma, o assistente de geração usa esses metadados para saber quais arquivos do ambiente de compilação e testes (**openbus/install**) formam um perfil de pacote. Esses metadados também são inseridos no pacote final gerado pelo assistente de geração. Assim, é possível que o instalador use esses metadados em casos (ainda não implementados) como:

1. saber quais desses arquivos são configurações e se já existirem na árvore de instalação, poder reconfigurá-los preservando as configurações anteriores
2. proceder a tarefas de desinstalação como observar quais arquivos existem numa árvore de instalação e decidir por apagar apenas os arquivos que originalmente faziam parte do pacote entregue ao cliente, sem apagar outros dados produzidos pela execução do software

O que não fazem?

Não há suporte a dependências entre pacotes

Para cumprir esse tipo de requisito é interessante fazer um merge entre o código atual dos assistentes com o sistema de empacotamento LuaRocks, que além de suportar dependências entre pacotes provém outras funcionalidades interessantes.

Não há suporte à plataforma Microsoft Windows

É possível ter esse suporte contanto que se mapeie os comandos de sistema demandados pelos assistentes em utilitários desse sistema operacional. No LuaRocks, por exemplo, é considerado o uso da ferramenta UnxUtils que parece uma ótima alternativa.