# Engenharia de Software : Assistente Compilação

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



## Organização de diretórios

Esse assistente considera que os seguintes diretórios existem com seus respectivos papéis:

- **openbus**
  - **lib** — código descompactado das bibliotecas externas que o OpenBus depende
    - **lua5.1**
    - **oil04**
    - *outras libs Lua*
    - **openssl-0.9.9**
    - **openldap-2.4.11**
    - *outras libs de sistema*
  - **packs** — pacotes obtidos do repositório e gerados pelo assistente de geração
    - **metadata** — arquivos intermediários gerados pelo assistente de compilação e usados na geração e instalação de pacotes
  - **trunk** — código-fonte mais atualizado
  - **OB_v1_10_00_2009_03_03** — outros branches devem estar no mesmo nível do trunk
  - **OB_v1_30_00_2009_03_10** — outros branches devem estar no mesmo nível do trunk
  - **install** — alvo das compilações para a versão mais atualizada dos fontes
  - **install-OB_v1_10** — alvo das compilações para um branch específico
  - **install-OB_v1_30** — alvo das compilações para um branch específico

## Dependências

As dependências marcadas como **ferramentas** devem estar previamente instaladas. Os **códigos-fonte das bibliotecas** podem ser obtidos do repositório de pacotes do OpenBus. O **código principal** do OpenBUS, por sua vez, encontra no subversion do TecGraf.

- Ferramentas:
  - Tecmake — ferramenta padrão no TecGraf para compilação multi-plataforma, usada tantos para os serviços básicos do OpenBus como para as bibliotecas baseadas em Lua.
  - Autotools — conjunto de ferramentas necessárias para compilação dos softwares externos (como OpenSSL e OpenLDAP
  - Ferramentas básicas de desenvolvimento da GNU (gcc, g++, ar, ranlib, make, etc)
  - Assistentes OpenBus para compilação/geração/instalação em sua forma binária
- Variáveis de ambiente:
  - **TEC_UNAME**, **TEC_SYSNAME** — definidas pelo scripts (tec_uname.csh ou tec_uname.bsh) do Tecmake que devem ser carregadas no .profile do usuário
  - **PATH**, **CPATH**, **LD_LIBRARY_PATH** — precisam estar atualizadas para encontrar as ferramentas de dependentes citadas acima
- Códigos-fonte:

- Bibliotecas Lua — serão descarregadas **automaticamente** [3] do repositório de pacotes do OpenBus
- Bibliotecas não-Lua — serão descarregados **automaticamente** [4] do repositório de pacotes do OpenBus
- OpenBus — código-fonte principal é descarregado **automaticamente** [5] do repositório subversion do TecGraf (engsoftware)

[3] *Atualmente temos um pacote fictício nomeado "lualibs" (descrito no packages.desc) que é útil **apenas** para obter e extrair os fontes das bibliotecas Lua.*

[4] *Precisa existir o comando **wget** ou **curl** (clientes http) para obter os pacotes do repositório.*

[5] *Precisa existir o comando **svn** instalado no sistema. Veja mais detalhes sobre configurações relacionadas ao uso do **svn**.*

# Passo-a-passo simplificado

Esse passo-a-passo considera uma organização de diretórios que pode ser personalizada.

1. Siga as instruções para instalação dos assistentes
2. Certifique-se que:
   a. Comando **svn** esteja disponível em sua instalação **ou** faça um *checkout* manual do repositório subversion do OpenBus
   b. As variáveis de ambiente (como PATH e LD_LIBRARY_PATH) estejam corretamente definidas e apontando para as **ferramentas** listadas como dependências para compilação
   c. (opcional) Caso o desenvolvedor prefira uma outra organização de diretórios deve configurar as variáveis internas dos assistentes para que reflitam seus caminhos preferenciais
3. Configure as variáveis de ambiente:
   a. Defina a variável de ambiente OPENBUS_HOME para o mesmo local definido na INSTALL.TOP do seu **tools/config.lua**
   b. Importe as configurações do arquivo **tools/shell/kshrc** (ou **tools/shell/cshrc**, se o shell for CSH)
      i. Isso definirá variáveis de ambiente importantes para compilação como LIBRARY_PATH e CPATH
      ii. Para facilitar mantenha tanto a definição da OPENBUS_HOME quanto a importação das configurações do shell no seu arquivo .profile ou .kshrc
4. Execute o comando:

```
tools compile
```

Especialmente na plataforma **SunOS510**:

1. Comando **env** está em local diferente, assim precisamos modificar a primeira linha dos arquivos compile.lua, installer.lua e makepack.lua para refletir o local adequado (para saber execute: **which env**).
2. É comum não termos o comando **gmake** disponível, então é preciso fazer um link simbólico (no seu PATH) do **make** para **gmake**.
3. Por outro lado, é preciso certificar-se que **não** se esteja usando o **make** nativo da solaris ou mesmo o **make gnu** anterior à versão 3.80. Essas versões não são compatíveis com o Tecmake e causam problema durante a compilação das bibliotecas Lua. O Tecmake permite informar pela variável **TECMAKE_MAKE** qual comando **make** será usado.

# FAQ

## Como o assistente identifica se o pacote precisa ser compilado?

O assistente de compilação tem a habilidade de verificar se é necessário compilar um determinado pacote. Esse procedimento baseia-se nos campos **test_libs** e **external_dependencies** contido nos arquivos de descrição (**.desc). Atualmente, apenas os pacotes das bibliotecas externas que usam *Autotools** permitem esse comportamento. Caso a descrição de um pacote definir algum desses campos será realizado:

1. verificação se todos os valores da **test_libs** estão presentes no sistema (baseado na variável de ambiente responsável pela localização de libs em cada plataforma). Caso alguma biblioteca esteja **ausente** o pacote em questão **será** compilado.
2. verificação se todos arquivos informados na **external_dependencies** estão presentes. Caso algum **não esteja disponível** será **impossível** compilar o pacote e assim retornará um erro.

## Como o assistente identifica se precisa fazer o download dos fontes de um pacote?

O campo **source** na descrição do pacote é opcional, mas se existir indica que a URL do repositório onde os fontes estão.

Após identificar se é necessário compilar o pacote, o assistente verifica se o pacote já foi obtido do repositório, verificando se já existe o arquivo no diretório **openbus/packs** (valor padrão da variável **DOWNLOADDIR**). Caso já tenha sido obtido o pacote será apenas extraído no diretório **openbus/lib** (valor padrão da variável **PRODAPP**). Caso já tenha sido extraído (teste da existência de um diretório com mesmo nome do pacote) então o assistente procede à compilação sem precisar extrair novamente.

## Como compilar apenas alguns pacotes?

O assistente *compile.lua* aceita o parâmetro **--select** que permite compilar apenas um pacote. Adicionalmente, esse parâmetro pode ser usado várias vezes. Exemplo:

```
tools compile --select="openssl-0.9.9 lua5.1" --select="openldap-2.4.11 openbus-core"
```

## Como listar quais pacotes serão compilados?

O assistente *compile* aceita o parâmetro **--list**, que pode ser combinado com os outros parâmetros vistos até aqui e apresentará em tela quais pacotes foram escolhidos e seriam compilados.

## Como recompilar um ou mais pacotes?

O assistente *compile* aceita o parâmetro **--rebuild** que executará **tecmake rebuild** ou **make clean**, a depender do método de compilação.

## Como e porquê forçar a compilação de pacotes?

O assistente *compile* aceita um importante parâmetro **--force**. Ao usar esse, as verificações do campo **test_libs** são desativadas em favor de tentar compilar todos os pacotes selecionados. Caso nenhum pacote tenha sido selecionado, assume-se que todos pacotes serão compilados. Esse uso permite-nos gerar um pacote de instalação mais completo, dependendo de menos softwares nativos no sistema operacional.

## Quais parâmetros são aceitos pelo assistente?

```
$ tools compile --help
[ CONSOLE ] Loading the assistant:      compile
 Usage: compile OPTIONS
 Valid OPTIONS:
        --help                  : show this help
        --verbose               : turn ON the VERBOSE mode (show the system commands)
        --basesoft=filename     : use the 'filename' as input for basic
                                  softwares with autotools semantic (i.e: openssl)
        --packages=filename     : use the 'filename' as input for packages
                                  with tecmake semantic (i.e: lua5.1, openbus-core)
        --rebuild               : changes the default rule for tecmake rebuild if
                                  already compiled
        --force                 : forces the compile and install (i.e: you want
                                  re-generate some library even it's installed
                                  already = to debug or devel purpose)
        --list                  : list all package names from description files. When
                                  '--select' is used, it'll confirm the selection.
```

```
        --select="pkg1 pkg2 ..." : choose which packages to compile and install
        --nosvn                  : don't try to checkout from svn

NOTES:
        The prefix '--' is optional in all options.
        So '--help' or '-help' or yet 'help' all are the same option.
```

## Como proceder ao compilar para várias plataformas?

O uso do **autotools** pelas bibliotecas externas não-Lua compila os códigos e mantém os produtos de compilação de forma que **é fundamental usar o assistente de compilação com o parâmetro --rebuild**. Exemplo:

1. considere que você tenha compilado o pacote **openssl-0.9.9** para a plataforma **Linux24g3**
2. agora você quer entrar na máquina **SunOS58** e compilá-lo também, logo
    a. é necessário usar o assistente com o parâmetro **rebuild** para limpar os objetos compilados antes de fazer uma nova compilação:

    ```
    $ tools compile select="openssl-0.9.9" rebuild
    ```

Situação semelhante ocorre com os pacotes que usam o **tecmake**. O Tecmake gera arquivos **.dep** com os objetos dependentes para a linkedição de uma biblioteca compartilhada (por exemplo). Logo, é importante ter cautela e analisar tais arquivos **.dep** em casos de problemas na compilação. Assim é recomendado o uso do **rebuild** ao trocar de plataforma de compilação.

## Como configurar o assistente de compilação para um conjunto diferente de diretórios?

Por padrão, os assistentes uma organização padrão de diretórios.

É possível alterar essa organização pela definição de variáveis globais num arquivo de configuração a ser informado aos assistentes. Todas as configurações possíveis podem ser observadas no código **openbus/trunk/tools/lua/tools/config.lua**. As **configurações do usuário que forem indicadas no momento da execução** dos assistentes **têm prioridade** sobre as configurações padrões.

Essas variáveis são importantes para o *compile*, *makepack* e *installer*. Abaixo as principais variáveis são descritas:

- TEC_UNAME : utiliza-se como valor a variável de ambiente de mesmo nome, baseado na nomenclatura do Tecmake.
- TEC_SYSNAME : utiliza-se como valor a variável de ambiente de mesmo nome, baseado na nomenclatura do Tecmake.
- BASEDIR : indica o diretório que armazenará a árvore de diretórios respeitando todos 3 ambientes.
    - Valor padrão **<home usuario>/openbus**.
- PRODAPP : indica o diretório que armazenará as bibliotecas externas que não se encontram no repositório SVN.
    - Valor padrão **BASEDIR/lib**.
- SVNURL : indica qual a URL completa do projeto no repositório SVN. Para compilar e gerar pacotes de outros branches é fundamental editar essa variável.
    - Valor padrão https://subversion.tecgraf.puc-rio.br/engsoftware/openbus/trunk.
- SVNDIR : indica qual o diretório que reflete um checkout do SVN. Para compilar e gerar pacotes de outros branches é fundamental editar essa variável.
    - Valor padrão **BASEDIR/trunk**. Exemplos para alguns branches: **BASEDIR/OB_v1_30_00_2009_03_10**, **BASEDIR/OB_v1_10_00_2009_03_03**.
- DEPLOYDIR : indica o diretório no qual os assistentes estão instalados. Para efeitos de bootstrap é fundamental assegurar-se que o valor dessa variável aponta para o local onde os assistentes foram instalados.
    - Valor padrão **SVNDIR/tools**.
- DOWNLOADDIR: indica o diretório que armazenará os .tar.gz obtidos durante a compilação e os gerados pelo assistente de geração de pacotes.
    - Valor padrão **BASEDIR/packs**.
- PKGDIR : indica o diretório que armazenará arquivos intermediários gerados pelo assistente de compilação e que são úteis ao assistente de empacotamento e instalação.
    - Valor padrão **DOWNLOADDIR/metadata/TEC_UNAME**.

- INSTALL.TOP : indica qual a pasta que será usada como ambiente de compilação. Essa variável é usada após a compilação via autotools ou tecmake como alvo das cópias do que for compilado.
  - Valor padrão **BASEDIR/install**. Exemplos para separar os ambientes de compilação para diferentes branches isolando-os: **BASEDIR/install-OB_v1_30**, **BASEDIR/install-OB_v1_10**.
- INSTALL.BIN : indica o diretório dentro do ambiente de compilação responsável por armazenar os binários.
  - Valor padrão **INSTALL.TOP/bin/TEC_UNAME**.
- INSTALL.LIB : indica o diretório dentro do ambiente de compilação responsável por armazenar as bibliotecas.
  - Valor padrão **INSTALL.TOP/libpath/TEC_UNAME**.
- INSTALL.INC : indica o diretório dentro do ambiente de compilação responsável por armazenar os cabeçalhos C/C++.
  - Valor padrão **INSTALL.TOP/incpath**
- TMPDIR : indica o diretório temporário usado como prefix para o autotools durante a compilação.
  - Valor padrão **/tmp/openbus-building_<random number>**

## Como compilar o barramento em um determinado branch?

Para compilar os fontes do OpenBus para um determinado branch é preciso usar um arquivo de configuração como entrada para o assistente semelhante a:

```
-- exemplo de uso do branch do barramento na versão 1.3
SVNURL = "https://subversion.tecgraf.puc-rio.br/engsoftware/openbus/branches/
OB_v1_30_00_2009_03_10"

BASEDIR = os.getenv("HOME").."/openbus"
SVNDIR = BASEDIR.."/OB_v1_30_00_2009_03_10"
INSTALL = { TOP = BASEDIR .."/install-OB_v1_30" }

-- tipicamente deve ser comum recompilar pacotes de um branch mas usando a versão mais nova
-- dos assistentes que:
--   ou podem estar instalados em algum local exclusivo
--   ou deve-se usar sua versão do trunk
-- como a DEPLOYDIR (por padrão) é definida baseada na SVNDIR, deve-se reconfigurá-la:
DEPLOYDIR = BASEDIR.."/trunk/tools"
```

A linha de comando do assistente fica semelhante a:

```
$ tools config=<arquivo> compile select=<nome do pacote> <outros argumentos>
```

É **fundamental** que além dessa configuração dos assistentes, a variável de ambiente **OPENBUS_HOME** esteja configurada no shell para apontar para o mesmo valor da **INSTALL.TOP**. Obviamente as variáveis de ambiente relacionadas ao valor da OPENBUS_HOME precisam ser atualizadas, por exemplo: **LD_LIBRARY_PATH**, **CPATH** e **PATH**. Todas as variáveis que precisam ser reconfiguradas são listadas nas dependências do assistente de compilação.

## Como definir um novo pacote para ser compilado?

Uma descrição simplificada de pacote precisa informar uma tabela Lua com os seguintes campos:

- **name** — string contendo a identificação do pacote
- **source** — string opcional, mas normalmente interessante, que indica a URL de onde baixar o fonte do pacote
- **build** — tabela com os seguintes campos:
  - **type** — string que identifica qual ferramenta de compilação deve ser usada. Possíveis valores: **autotools**, **tecmake** ou **copy**. Para outros métodos bastar criar um arquivo .lua na pasta **trunk/tools/build** com o respectivo nome e definir uma função **run** que saiba manipular o descritor.
- **install_files** — tabela contendo <chave,valor> (ambos strings) onde a *chave* define **de onde** os arquivos ou pastas devem ser copiados (pode-se usar variáveis definidas no **trunk/tools/config.lua**, como TMPDIR)
- **dev_files** — tabela semelhante a **install_files** mas que na etapa de geração do pacote irá compor pacotes nomeados **<nomepacote>-dev**. Útil para separar pacotes para *administradores* e para *desenvolvedores*.

- **conf_files** — tabela semelhante a **install_files** que irá compor pacotes nomeados **<nomepacote>-conf**. Útil para separar as configurações (que podem ser personalizadas pelo usuário) dos binários e bibliotecas.
- **conf_template** — string que indica qual arquivo de template será usado para fazer perguntas sobre a configuração do pacote

Os seguintes campos da tabela **build** dependem de cada tipo:

- **autotools** permite:
  - **TEC_UNAME** — podem haver vários campos com diferentes definições por plataforma, deve-se preencher o nome do campo diretamente com o TEC_UNAME ou TEC_SYSNAME desejados. Durante a compilação (numa plataforma) se não houver o TEC_UNAME será verificado se há um campo com TEC_SYSNAME, caso não exista será emitido um erro.
  - **test_libs** — tabela com strings com padrões de nomes para as bibliotecas que representam o pacote. Útil para saber se o pacote já está instalado ou não.
  - **external_dependencies** — tabela contendo chaves:
    - **bins** — string com o nome ou padrão dos binários necessários para compilar o pacote em questão
    - **libs** — string com o nome ou padrão das bibliotecas necessárias para compilar o pacote em questão
    - **includes** — string com o nome ou padrão dos cabeçalhos necessários para compilar o pacote
- **tecmake** permite:
  - **src** — string contendo o caminho do diretório para os fontes, de onde será executado o comando *tecmake*
  - **mf** — tabela com strings contendo nomes dos arquivos .mak. Se existir **config.mak** basta indicar **config**. Isso vai produzir a execução do comando **tecmake MF=config**. Algumas dicas são muito úteis:
  - Se for necessário passar outras flags de compilação também é possível, como por exemplo *config LUA51=caminho_instalacao_lua* que executará o comando **tecmake MF=config LUA51=...**, ou ainda **server genstubs** que executará **tecmake MF=server genstubs**.
  - Caso para a geração dos binários ou bibliotecas de um único pacote existam mais de 1 arquivo .mak, é possível passar mais de um valor na tabela **mf**, por exemplo: **mf = { "config LUA51=/caminho", "server genstubs", "server" }**.

Os seguintes exemplos ilustram a descrição de um pacote para cada método de compilação:

- **Autotools**:

```
{ name = "cyrus-sasl2-2.1.22.dfsg1",
  source = "http://www.tecgraf.puc-rio.br/~amadeu/tarballs/cyrus-sasl2-2.1.22.dfsg1.tar.gz",
  build = {
    type = "autotools",
    test_libs = {"libsasl2*.*"},
    external_dependencies = {
        includes = "db.h",
        libs = "libdb.so",
    },
    Darwin = "./configure --prefix=".. TMPDIR .." --disable-macos-framework --disable-ldap --
enable-static=yes"..
        " && cd include/ && make && make install && cd ../lib/ && make && make install ",
    Linux =  "./configure --prefix=".. TMPDIR .." --disable-ldap --enable-static=yes"..
        " && cd include/ && make && make install && cd ../lib/ && make && make install ",
    SunOS510x86 =  "env CC=gcc ./configure --prefix=".. TMPDIR .." --disable-ldap --enable-
static=yes"..
        " && cd include/ && make && make install && cd ../lib/ && make && make install ",
  },
  install_files = {
    [TMPDIR .."/lib/libsasl2*.*so*"] = "libpath/${TEC_UNAME}",   --dynamic linux,aix,sunos
    [TMPDIR .."/lib/libsasl2*.*dylib*"] = "libpath/${TEC_UNAME}",--dynamic macos
  },
  dev_files = {
    [TMPDIR .."/lib/libsasl2*.a*"] = "libpath/${TEC_UNAME}",    --static linux,aix,sunos
    [TMPDIR .."/lib/libsasl2*.*so*"] = "libpath/${TEC_UNAME}",   --dynamic linux,aix,sunos
    [TMPDIR .."/lib/libsasl2*.*dylib*"] = "libpath/${TEC_UNAME}",--dynamic macos
    [TMPDIR .."/include/sasl"] = "incpath/cyrus-sasl2-2.1.22.dfsg1",
```

```
  },
}
```

- **Tecmake**:

```
{ name = "openbus-core",
  build = {
    type = "tecmake",
    src = SVNDIR .."/core/services",
    mf = { "config LUA51="..PRODAPP.."/lib/lua5.1", },
  },
  install_files = {
    ["../bin/${TEC_UNAME}/*"] = "core/bin/${TEC_UNAME}",
    ["../bin/*.sh"] = "core/bin",
    ["../bin/servicelauncher"] = "core/bin",
    ["../idl/*"] = "idlpath",
    ["../services/accesscontrol"] = "core/services",
    ["../services/registry"] = "core/services",
    ["../services/session"] = "core/services",
    ["../utilities/lua/openbus"] = "core/utilities/lua",
  },
  conf_files = {
    ["../../data/conf"] = "data",
    ["../../data/certificates"] = "data",
  },
  conf_template = DEPLOYDIR .. "/templates/openbus.lua",
  dev_files = {
    ["../test/lua"] = "core/test",
  },
}
```

## Como o assistente de compilação reconhece as novas descrições de pacotes?

O assistente *compile*, por padrão, utiliza os arquivos **tools/basesoft.desc** e **tools/packages.desc**. Em ambos arquivos as descrições devem estar na ordem que serão compiladas, então se um pacote A precisa que o pacote B já esteja compilado, é preciso que no arquivo .desc B apareça antes de A. O arquivo **basesoft.desc** contém as descrições das bibliotecas externas (não Lua) que normalmente usam os métodos **autotools** ou **copy**, a exemplo da OpenSSL e OpenLDAP. O arquivo **packages.desc** contém as descrições das bibliotecas Lua e dos códigos principais do OpenBus que usam o método **tecmake**.

Os nomes (ou localizações) desses arquivos .desc podem ser alterados contanto que então se utilize as opções da linha de comando para informá-los:

```
tools compile --basesoft=/meurepositorio/basesoft.desc --packages=/meurepositorio/packages.desc
```

Uma proposta de melhoria está reportada no OPENBUS-123.