Setting up an SDK for Secondo

Database Systems for new Applications
University of Hagen
http://www.informatik.fernuni-hagen.de/secondo

Last Changes: 11/6/06

Contents

1 Introduction

This guide explains the installation procedure for Secondo on computers running the Linux operating system. Secondo depends on some software and therefore we provide some insights on how to install them too.

This guide explains the installation procedure for various software packages needed to set up an complete development environment ready to compile Secondo. We have built the system on the following operating-systems:

- SuSe Linux: 8.1, 8.2, 9.0, 9.2, 10.x
- Mac-OSX: Tiger
- MS-Windows: Win98, Win2000, WinXP

Secondo's source code has the following dependencies:

Tool/Compiler	Version	Home page	CD-Directory
bash	2.05b	www.gnu.org	extras/unix
make	3.79.1	www.gnu.org	extras/unix
gcc/c++	3.4.6	gcc.gnu.org	linux/gnu
bison	2.1	www.gnu.org	linux/gnu
lib readline	4.3	www.gnu.org	linux/gnu
lib ncurses	5.3	www.gnu.org	linux/gnu
findutils	4.2.25	www.gnu.org	linux/gnu
flex	2.5.33	www.gnu.org	linux/non-gnu
lib jpeg	v6b	www.ijg.org	linux/non-gnu
berkeley-db	4.2.52	www.sleepycat.com	linux/non-gnu
swi-prolog	5.4.7	www.swi-prolog.org	linux/prolog
java2 sdk	1.4.2	www.javasoft.com	linux/j2sdk

On the Secondo home page we have collected them into Installation-Kits (Linux, Windows specific versions without Java2-SDK) and an ISO image of a CD-ROM which contains all tools for both operating systems and some databases. All tools with a GNU license can be found via www.gnu.org, there is a link to the free software directory (http://directory.fsf.org). This is a portal to all known free software.

However, you can try to get the needed tools from the Internet and set up your own environment. To some extent it may also work with higher or smaller version numbers as recommend above. But we cannot guarantee that it will work. The version above is the setup for which we know that it works. Our experience is that even small changes in versions may cause compile time or runtime errors.

We follow three approaches to install the required 3rd-party software (i) the installation script, (ii) instructions for compiling the packages itself, (iii) instructions for installing RPM-packages. The last two alternatives are not possible for MS-Windows. For this platform many pre-compiled files can be found at http://sourceforge.win32.net and www.mingw.org. .

The first one assumes that you download one of the Installation-Kits or that you got a CD-ROM. These provide a script installsdk which will uncompress, compile and install the above tools with minimal user interaction below your `$HOME` directory.

The second approach is to download the source code from their corresponding web sites, compile, and install them. With this approach, it is possible to perform the installation steps as an ordinary user. It is just to do the steps of (i) manually. This information may help you if the script failed in building some of the packages, or if you want to implement algebras which need features not covered by the standard SDK.

The third one is to use a Linux distribution package installer. Some people will prefer this approach because the package installers normally check dependencies between software. Given that we are experienced on using SuSE and Linux, we provide installation instructions for both using the package administration tool YaST. In fact we do not provide information on how to

use the YaST tool, but only the package names needed. To do this you need to use the super user account *root*. But keep in mind, that Secondo's source code may not be suitable for the versions which you will install by this way. Hence it is a very experimental approach.

Throughout this guide, we assume that the *bash* is used as shell and some base software are already installed, e.g. *tar*, a C compiler *gcc*, and *make*.

2 Running the script `installsdk`

Download the installation kit of the secondo home page and extract it

```
tar -xvzf <inst-kit>.tar.gz
```

Alternatively you may have got a CD which contains all the needed stuff. Then copy it to your disk into a separate directory. Now go into the new created directory and run the script, e.g.

```
cd <inst-kit-dir>
./installsdk -p<platform>
```

The script will create by default a subtree under `$HOME/secondo-sdk` where it installs all 3rd-party software. The script nows the following options:

```
 -h print this message and exit.
 -d name of the the SDK root directory [default = secondo-sdk].
 -p platform name; one out of {linux, win32, mac_osx}.
 -s Use the system's gcc, don't try to install the provided one.
```

The compilation of packages is done below the `/tmp/secondo-tmp-<USER>/secondo-sdk` directory, the script will report the exact destination. Alternatively you may choose a destination. In the following we will refer to this directory by the variable `SECONDO_SDK`.

During the installation two child windows appear. The first contains the Java installation which needs some user interaction, and the other displays the messages of the compiler during installation. All tools will be installed below the directory `$HOME/secondo-sdk`, it will create the following subdirectories:

```
/
|---/home/$USER
            |---secondo
            |---$SECONDO_SDK
                        |--- gcc
                        |--- aux
                        |--- flex
                        |--- bison
                        |--- bdb
                        |--- swi
```

However, we can not guarantee, that the script works on all available Linux versions, but we hope that it basically works and helps to save time.

If some steps fail, you may need to study the log file and correct the installation of the failed packages by hand. Therefore you have to change into the build directory of the package and to run make distclean. Afterwards you can try to configure it with other options or you may download another version. However, in this case you have to read the package's installation notes.

The most critical part of the installation is that of the *gcc*. The problem is, that if the version of your system's gcc and the one to be installed is highly different this may result in an compile error. If you cannot manage to install the provided gcc, then you can try the `-s` switch. With this option, the system's gcc will be used.

At the end, the script will create some shell scripts which basically set up a lot of environment variables which are needed at compile time and runtime in your `$SECONDO_SDK` directory, these are

```
secondorc                  // the basic inititalization script
secondo.aliases            // some alias definitions
secondo.config.$platform   // optional settings
secondo.setroot            // definition of important search path lists
libutil.sh                 // a library of useful shell functions
```

The `*.config.*` file needs to be changed if some tools are not installed in their assumed default installation directory, e.g. JAVA2-SDK, if it is already present. In order to execute the commands in the current instance of the shell you need to run

```
export SECONDO_SDK=$HOME/secondo-sdk
export SECONDO_PLATFORM="linux" # for Mac-OSX use value "mac_osx" instead
export SECONDO_BUILD_DIR=$HOME/secondo
source $SECONDO_SDK/secondorc
```

You can also add this command in the file `~/.bashrc` which will be executed automatically when a new shell is invoked. Moreover, the following commands aliases are available:

- `secinit`: Sets some environment variables to some default values (it simply executes `secondorc`). Afterwards your shell environment will be altered and it should be possible to compile and link Secondo.
- `secenv`: Displays the values of some important environment variables.
- `secroot`: Defines the current directory as root of a Secondo source tree (changing variable `SECONDO_BUILD_DIR`). This is useful if you have different source trees and want to change between them.

On MS-Windows. Most of the tools will be installed as binaries. Only the Berkeley-DB will be compiled. The gcc compiler will be installed with MSYS/MinGW, thus the created directories are:

```
C:
 |---\secondo-sdk
 |         |--- aux
 |         |--- flex
 |         |--- bison
 |         |--- bdb
 |         |--- swi
 |
 |---\msys\1.0\home\$USER
 |                     |-.bashrc
 |                     |---secondo
 |---\mingw
```

3 Compiling all the packages

In this approach, all software will be compiled and installed in the system. For that, the tools tar, make, and gcc must be previously installed. One can check if they are present in the system running the following commands:

```
tar --version
make --version
gcc --version
```

If one of them is not present, please contact the system administrator to install them. In most Linux installations, these applications are already available.

Depending on the system, some of the packages needed are already installed. One could first check whether they are installed before really installing them. Skip the steps to install the packages that are already installed in your system.

In order to compile and install all the packages, we suggest to create a directory in the user home directory called secondo-sdk, so that it is possible to perform the installation steps as an ordinary user. If one has superuser access and wants to put the software available for all users, another directory can be used, e.g. `/usr/local` (see the –prefix argument in the configure step). We also suggest the creation of a temporary installation directory called secondo-install, which can be removed after the end of the process. We will also create environmental variables to point to these two directories, namely `SECONDO_SDK` and `SECONDO_INSTALL`, respectively. Enter the following commands:

```
cd $HOME
mkdir secondo-sdk
export SECONDO_SDK=$HOME/secondo-sdk
mkdir secondo-install
export SECONDO_INSTALL=$HOME/secondo-install
```

Let us also add the *bin* directory of secondo-sdk to the PATH and the *lib* directory to the `LD_LIBRARY_PATH`

```
mkdir $SECONDO_SDK/bin
export PATH=$SECONDO_SDK/bin:$PATH
mkdir $SECONDO_SDK/lib
export LD_LIBRARY_PATH=$SECONDO_SDK/lib:$LD_LIBRARY_PATH
```

Normally, compiling and installing software is done in four steps:

- downloading the source and decompressing it,
- calling the configure script to check system defaults and to set some options. The most important option is `--prefix` which defines the root of a subtree where the files should be installed.
- running `make`, which compiles the sources, and
- running `make install` to copy the files to their destination.

3.1 Building the C++ compiler

In some Linux systems, only the C compiler comes pre-installed but not the C++. As said before, we are assuming that the C compiler is already installed.

First, it is good to check it the C++ compiler is already installed in the system. The following command will print information about it.

```
g++ --version
```

In the following we will configure for sub-directories below `$SECONDO_SDK`. This make life easier if you want to delete a package or to install another alternative version of a package. As a drawback you need to take more care about search path configuration. We will use the same directory structure as `installsdk` does (refer to section 2).

If the compiler is not installed, then we have to build it. Otherwise skip the next sub-sections. The only dependency for the C++ compiler is the *TeX-info* package, which itself depends on the *ncurses* library.

3.1.1 Building the library *ncurses* without C++ support

At this point we do not have a C++ compiler, then we will build this package without support for C++.

Download the file `ncurses<version>.tar.gz` into the secondo-install directory. Run the following set of commands.

```
cd $SECONDO_INSTALL
tar zxvf ncurses<version>.tar.gz
cd ncurses<version>
./configure --prefix=$SECONDO_SDK/aux --disable-nls --without-cxx-binding
make
make install
```

3.1.2 Building the package *TeX-info*

We have found a problem while configuring the *TeX-info* package that it seems not to recognize the *ncurses* library. We then run this command in order to tell it where is the *ncurses* library.

```
export LDFLAGS="-L$SECONDO_SDK/aux/lib -lncurses"
```

Now we can build the *TeX-info* package. Download it into the secondo-install directory and run the following commands.

```
cd $SECONDO_INSTALL
tar zxvf texinfo<version>.tar.gz
cd texinfo<version>
./configure --prefix=$SECONDO_SDK/aux --disable-nls
make
make install
```

3.1.3 Building the C++ compiler

Download the *gcc* package from gcc.gnu.org into the secondo-install directory and execute the following commands

```
cd $SECONDO_INSTALL
tar zxvf gcc<version>.tar.gz
mkdir gcc<version>.objs
```

The *gcc* documentation strongly recommends that we create a directory for the object files outside the directory tree of the *gcc*.

```
cd gcc<version>.objs
../gcc<version>/configure --prefix=$SECONDO_SDK/gcc \
                          --disable-nls --enable-bootstrap
make bootstrap
make install
```

3.2 Building the library *ncurses*

Even if you followed the procedure in Section 3.1.1, you have only created the *ncurses* library without support for C++.

If you did not follow the procedure in Section 3.1.1, then download the library *ncurses* into the secondo-install directory, and run the following set of commands.

```
cd $SECONDO_INSTALL
tar zxvf ncurses<version>.tar.gz
```

Now we can build the *ncurses* library with the commands:

```
cd $SECONDO_INSTALL
cd ncurses<version>
./configure --prefix=$SECONDO_SDK/aux --disable-nls --with-cxx-binding
make
make install
```

3.3 Building the library *readline*

Download the library into the secondo-install directory and run the following commands:

```
cd $SECONDO_INSTALL
tar zxvf readline<version>.tar.gz
cd readline<version>
./configure --prefix=$SECONDO_SDK/aux --with-curses
make
make install
```

3.4 Building the *bison* package

Download the package into the secondo-install directory and run the following commands:

```
cd $SECONDO_INSTALL
tar zxvf bison<version>.tar.gz
cd bison<version>
./configure --prefix=$SECONDO_SDK/bison
make
make install
```

3.5 Building the *flex* package

Download the package into the secondo-install directory and run the following commands:

```
cd $SECONDO_INSTALL
tar zxvf flex<version>.tar.gz
cd flex<version>
./configure --prefix=$SECONDO_SDK/flex
make
make install
```

3.6 Building the *jpeg* library

One should note that the *jpeg* library is only necessary if the Picture Algebra is used. Otherwise this section can be skipped.

Download the package into the secondo-install directory and run the following commands:

```
cd $SECONDO_INSTALL
tar zxvf jpegsrc<version>.tar.gz
cd jpegsrc<version>
./configure --prefix=$SECONDO_SDK/aux
make
make install
make install-lib
```

3.7 Building *Berkeley DB*

Download the package into the secondo-install directory. We suggest downloading the version without encryption (the file name contains `NC`), since Secondo does not use this feature.

Run the following commands:

```
cd $SECONDO_INSTALL
tar zxvf db<version>.NC.tar.gz
cd db<version>.NC/build_unix
../dist/configure --prefix=$SECONDO_SDK/bdb --disable-nls
make
make install
```

3.8 Building *SWI-Prolog*

Download the package into the secondo-install directory. We suggest downloading the version without encryption (the file name contains `NC`), since Secondo does not use this feature.

Run the following commands:

```
cd $SECONDO_INSTALL
tar zxvf pl-<version>.tar.gz
cd pl-<version>
../dist/configure --prefix=$SECONDO_SDK/swi
make
make install
```

3.9 Building the *java* package

This package cannot be built by compiling it, Since it is no free software (see also www.gnu.org/philosophy/java-trap.html). One has to download an executable file called `jdk<version>.bin` into the secondo-install directory and execute it from the secondo-sdk directory as follows:

```
cd $SECONDO_SDK
$SECONDO_INSTALL/jdk<version>.bin
```

4 Installing the packages with YaST under SuSE Linux

Here is the list of all packages that must be installed. The search field inside YaST can be used to filter packages by name. If a package is already installed, just leave it marked. If YaST presents other dependencies that are not provided here, just let it install them.

Some packages, e.g. java and swi-prolog, are not provided in the standard installation of SuSE, because they are not free software. One must use the additional installation CD (or repository) in order to install such packages.

4.1 Package *make*

```
make
```

4.2 Package *tar*

```
tar
```

4.3 Package *binutils*

```
binutils
```

4.4 Package *gcc* with support to C++

```
gcc
libstdc++-devel
gcc-c++
```

4.5 Package *bison*

```
bison
```

4.6 Package *flex*

```
flex
```

4.7 Package *java*

```
java-<version>-sun
jpackage-utils
update-alternatives
java-<version>-sun-devel
```

4.8 Library *jpeg*

```
libjpeg
libjpeg-devel
```

4.9 Library *readline*

```
readline
readline-devel
```

4.10 Library *ncurses*

```
ncurses
ncurses-devel
```

4.11 Package *swi-prolog*

```
gmp
gmp-devel
swi-prolog
```

4.12 Package *Berkeley DB*

```
db
db-devel
db-utils
```

5 Environment Setup

Use the configuration files provided in the `CM-Scripts` directory of the secondo source code. Copy the files `secondo*` and `libutils.sh` which are located in `<inst-kit>/scripts` into your `SECONDO_SDK` directory. The files for defining alternative paths are

```
secondo.config.linux
secondo.setroot
```

You should study them and change the variable definitions to your needs. Afterwards run

```
export SECONDO_SDK=$HOME/secondo-sdk
export SECONDO_PLATFORM=linux
export SECONDO_BUILD_DIR=$HOME/secondo
source $SECONDO_SDK/secondorc
```

you can check the Secondo related variables by calling the command `secenv`.

Now proceed with the sections 4 and 5.

6 Compiling Secondo

Start a new shell and make sure that the environment variables explained in section 4 are set.

Change to the directory which contains the secondo source code and run make, e.g.

```
cd $HOME/secondo
make
```

Afterwards you should run

```
make runtests
```

which runs some automatic tests to be sure that everything works fine. If something fails please send a bug report to `secondo@fernuni-hagen.de`

7 Installing Latex and Ghost-script

Secondo's program code documentation is made with the PD-System, which translates program code files into LaTeX. Additionally, for previewing EPS pictures together with LaTeX you need Ghost-script with its preview program. Hence, for using the PD-System you have to install this software using the system administration tools provided with your Linux system.

8 Installation Hints for other Unix based Systems

Apart from the SuSe Linux releases, other Unix like systems are conceivable, but we have no experience in compiling and installing the development environment on those systems. However, since all packages are compiled during installation it should work. A problem could be the Java installation since this is a binary version for i686-processors which will not run on other processor architectures like power-pc or SPARC.

Another problem could be that a commercial Unix derivate provided from SUN, HP, DEC etc. does not have the gnu tools instead there are vendor specific versions of the command shell, the C-Compiler, the Linker, the Assembler, etc. Hence we recommend the following steps:

- If there is no gnu bash on your system, you will find it in the `<Inst-kit>/extras/unix` directory. Install also the gnu `binutils` package into the directory `$HOME/secondo-sdk`.
- Extract the gcc packages and read carefully the installation manual of the gcc. In the configure step use options which define the linker and assembler in order to be sure that the tools of the binutils package are used. Install the gcc also into directory `$HOME/secondo-sdk` and set up your environment such that this gcc will be used. Verify it by calling `gcc --version`.
- Run the script with the -s option (see section 2) and follow the instructions as explained for Linux.