Dklibs installation on Windows

Dipl.-Ing. D. Krause

July 13, 2006

Contents

1 Overview

This manual tries to give you hints to install dklibs and some of the applications based on these libraries on a Windows system.

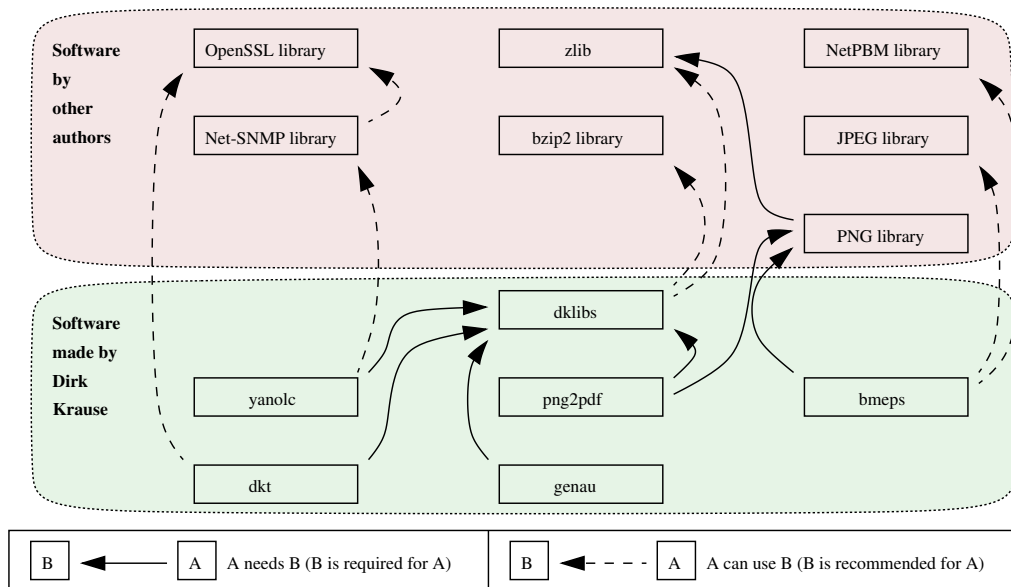Figure 1 shows the dependencies between the packages.

Figure 1: Software dependencies

2 Software and skills needed for the build

To build the software you need an ANSI-C-compiler, a linker and a "make" program. There are several development suites available, both commercial software (MS Visual Studio, Borland/Imprise products) and open-source software (Cygwin, MingW).
To build OpenSSL from source you also need a working Perl distribution. I suggest to use ActivePerl [1]. An assembler is helpful to build OpenSSL, you can use either MASM (available via platform SDK) or NASM.
You should have a working knowledge about:

- the command line options of the C compiler, linker and make program you use and

- the structure of makefiles.

---

[1]see http://www.activestate.com

3 Some decisions before starting to install

Before you start any build process you have to decide whether you want to have statically linked software or dynamically linked software.
I recommend static linking.
Using MS Visual C you have several versions of the C run-time library available. The preferred run-time library file for each source module is choosen by *compiler* switches when compiling the module.

Table 1: Command line switches to choose run-time library

Switch	File
/ML	LIBC.LIB (static, release, single-threaded)
/MLd	LIBCD.LIB (static, debug, single-threaded)
/MT	LIBCMT.LIB (static, release, multi-threaded)
/MTd	LIBCMTD.LIB (static, debug, multi-threaded)
/MD	MSVCRT.LIB (DLL, release)
/MDd	MSVCRTD.LIB (DLL, debug)

Object modules contain an information which run-time library they want to be linked with. During linking the linker inspects all object files – even those in libraries – and tries to use all the run-time libraries mentioned in any object module. If two (or more) modules were compiled specifying a different run-time library the linker will report conflicts and abort.
Conclusion: You need to use the same run-time library selection switch for all objects (both for installing the required/recommended libraries and installing dklibs and software based on it).
I recommend to use "/MT" for static linking or "/MD" for DLL-linking.
As pointed out you need to use the same run-time selection switch for all packages to build. When building DLL verions of the required/recommended libraries our compiler switches might differ from the compiler switches used to build the "official" DLL. If you have other software installed built to use one of the "official" DLLs strange things might happen if this software finds our self-made DLLs instead. That's why I recommend to use static linking.

4 Installing the required/recommended libraries

4.1 Installation order

I suggest to build the libraries in the following order:

- zlib
  http://www.gzip.org/zlib
  This library can be used by the dklibs library set to access gzip compressed files. If you want to build bmeps or png2pdf this library is required.

- bzip2
  http://sources.redhat.com/bzip2
  This library can be used by the dklibs library set to access bzip2 compressed files.

- libpng
  http://www.libpng.org
  This library is used by bmeps and png2pdf to read PNG images. It is required to build these packages.

- JPEG lib
  ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz This library can used by bmeps to read JPEG files, it is optional.

- NetPBM lib
  ftp://ftp.metalab.unc.edu/pub/Linux/apps/graphics/convert/netpbm-10.11.10.tgz
  This library can be used by bmeps to read NetPBM images, it is optional. The NetPBM tools allow to convert a wide variety of bitmap graphics formats to the NetPBM formats.

- OpenSSL
  http://www.openssl.org
  This library can be used by the kls application to calculate and print message digests (checksums) for files, it is optional. The Net-SNMP library can optionally use OpenSLL for authentication.

- Net-SNMP
  http://net-snmp.sourceforge.net
  This library can optionally be used by the yanolc package to build the snmpyalc program to request printer status information via SNMP. If none of your printers has a direct ethernet connection via internal print servers (i.e. HP JetDirect interface) the snmpyalc will probably not be usefull for you.

5

It is not in all cases necessary to install all libraries, see figure 1 on page 2 for package dependencies.

4.2 General instructions

4.2.1 Makefile modifications

The library archives contain makefiles named "makefile.msc" or "makefile.vc" for use with MS Visual C++. Additionally there are build instructions in readme files, file names differ.
To build the libraries follow the instructions in the readme files, but before running "nmake" or "make" modify the makefile manually:

- Open the makefile in a text editor,

- Search for a section introduced by a line

      .c.obj:

- The line after ".c.obj:" specifies the command to invoke the C compiler. You need to remove any command line switch choosing a C run-time library version and replace it by the switch you have decided to use in all the builds.

- Sometimes the command line switches are not specified directly, a macro expansion mechanism is used instead.
  If the command line looks like

      CL $(CFLAGS) /c $*.c

  you have to search for a line started by "CFLAGS=" to do the replacements there.

4.2.2 Building the libraries

To build the libraries run

    nmake −f makefile.msc

The name of the makefile may differ. For some of the libraries it is usefull to give the name of the library to build on the command line, i.e.

    nmake −f makefile.msc libbz2.lib

4.2.3 Copying files in place

If you are doing static builds, create a directory "c:\p\lib-stt", for a DLL-based build use "c:\p\lib-dll". Create subdirectories "lib", "include" and "bin". Do the following copy operations

```
xcopy  *.h   c:\p\lib-stt\include\  /Y
xcopy  *.lib  c:\p\lib-stt\lib\      /Y
xcopy  *.exe  c:\p\lib-stt\bin\      /Y
```

For a DLL-based build this loooks like

```
xcopy  *.h   c:\p\lib-dll\include\  /Y
xcopy  *.lib  c:\p\lib-dll\lib\      /Y
xcopy  *.dll  c:\p\lib-dll\bin\      /Y
xcopy  *.exe  c:\p\lib-dll\bin\      /Y
```

4.3   Library-specific instructions

4.3.1   LibPNG

The makefile.msc assumes to find the zlib library in "../zlib". We have installed zlib in subdirectories of "C\p\lib-stt" so we need to correct the makefile entry to "-I"C:\p\lib-stt\include"". For a DLL-build replace "lib-stt" by "lib-dll".

4.3.2   JPEG library

The JPEG library is shipped without a *.def file for DLL building. The contrib/jpeg directory in bmeps contains a "jpeg.def" file.

4.3.3 NetPBM library

The NetPBM library is shipped without a *.def file too. The contrib/netpbm directory in bmeps contains a "netpbm.def" file.
To build the NetPBM library unpack the sources archive and go into the "lib" subdirectory. Copy all the files from the "util" subdirectory into the "lib" directory. Edit the "shhopt.c" file, add the function

```c
char *rindex(char *s, int c)
{
    char *back = NULL;
    char *ptr;
    ptr = s;
    while(*ptr) {
     if(*ptr == c) {
      back = ptr;
     }
     ptr++;
    }
    return back;
}
```

Edit the file "libpm.c" and add the following two sections:

```c
#include <io.h>
#include <process.h>
#include <stdlib.h>

static int _S_ISREG(int m)
{
  int back = 0;
  if((m & _S_IFMT) == _S_IFREG) {
    back = 1;
  }
  return back;
}
```

4.3.4 OpenSSL

The makefiles "nt.mak" and "ntdll.mak" are generated (or possibly renewed) by the

```
perl Configure VC–WIN32
ms\do_masm
```

commands. You have to edit the makefiles to set up the run-time library version after running the two commands above and before running

```
nmake −f ms\nt.mak
```

or

```
nmake −f ms\ntdll.mak
```

Additionally remove all the optimization/warning related compiler flags and add "-DOPENSSL_NO_HW_PADLOCK=1 /Zm1000" to increase the used memory pool for the compiler and to disable inline-assembling of eng_padlock.c.
For the static library we should have

```
CFLAGS= /MT /nologo −DOPENSSL_SYSNAME_WIN32 ... \
   −DOPENSSL_NO_KRB5 −DOPENSSL_NO_HW_PADLOCK=1 /Zm1000
```

4.3.5 Net-SNMP

The Net-SNMP library archive contains a project/workspace for MSVC. When following the instructions in the readme you will possibly add OpenSLL support. Among other things you need to add the libraries "libeay32.lib" and "ssleay32.lib" to all the projects. Make sure that "gdi32.lib" is also included. Newer version of Net-SNMP use the "gdi32.lib" library (I do not know exactly but I think it is to get entropy from the screen). Somewhere in the project settings you can change the setting for the C run-time. Switch this to the library you have decided to use for all the projects.

5 Dklibs and applications

You have to install the dklibs library set before you try bo build any application using it. There is no specific order to build the applications, choose as you want. The archives contain a file named "makefile.msc". Each of these makefiles has two configuration sections possibly needing modifications.

- The first configuration section is at the beginning of the file. This section is used for

  - locations of directories and
  - subsections to indicate which libraries are available.

  The subsections for required libraries are ready-to-use. Subsections for recommended libraries must be toggled to use the library. Toggling such a subsection means to add a comment sign ("#") in front of all lines not started by "#". For all lines started by "#" remove the leading "#".
  Note: a subsection ends at an empty line.

- The second configuration section contains information how to build libraries: either DLL-linked or statically linked. The default is static linking, toggle all the subsection to switch to DLL-linking.
  Switching to DLL-linking also requires to toggle some lines in the first configuration section to activate the installation commands to copy *.DLL files into the right places. For zlib we also need to correct the name of the linker import library.