
Subject: Anyone using IDL_Tools DLM on Mac OS X?

Posted by [dick](#) on Wed, 12 Feb 2003 20:18:58 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi all,

Many of you know of the excellent set of routines available in the IDL_Tools DLM written by Randall Frank, described and available at: <http://www.kilvarock.com/freesoftware/dlms/randallfrank.htm>

The source code is there, as well as the binaries for Windows, but I will soon be needing binaries for Mac OS X (mainly for the Sockets routines). Has anyone out there managed to get this running? Or, would any kind soul be willing to look at the Makefile (attached below for convenience) and say whether this is an easy task?

I don't have an OS X machine to work with yet, but I may need to get this running quickly when the time comes to install on the client's machine.

Any help is much appreciated.

Cheers,

--

-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / <http://www.d-jackson.com>
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

----- Makefile -----

```
#
# Unix makefile for IDL_TOOLS DLM.
# $Id: Exp$
#

# The following macro might need to be edited if the location
# of IDL is not the standard location on your system.
#IDL_DIR = /data/rsi/idl_5.2

# The following macros establish general defaults. They are overridden
# in the "all" rule as needed to satisfy a given platform's
# requirements.

CC = cc
C_FLAGS = -I$(IDL_DIR)/external -c $(CFLAGS)
```

```
LD = ld
SHELL = /bin/sh
X_CFLAGS =
X_LD_FLAGS =
SO_EXT =so
```

```
.c.o :
$(CC) $(C_FLAGS) $(X_CFLAGS) $*.c
```

```
# The following is the default entry point. This section will
determine
# what system we are on and call make again with the correct flags
set.
```

```
all :
@echo "OS type detected: "`uname`
@if [ ${#IDL_DIR} = 0 ]; then \
echo "IDL_DIR variable must be set for DLMs" ; \
else \
case `uname` in \
"SunOS") if [ -d /proc ]; then \
make idl_tools \
"X_CFLAGS=-K pic" \
"X_LD_FLAGS=-B symbolic -z nodefs -G" ; \
else \
make idl_tools "CC=acc" \
"X_CFLAGS= -pic" \
"X_LD_FLAGS= -assert pure-text" ; \
fi \
;; \
"AIX") make idl_tools \
"LD=cc" \
"SO_EXT=a" \
"X_CFLAGS=-qro -qroconst" \
"X_LD_FLAGS= -bM:SRE -bnoentry -btextro -bE:idl_tools.export
-bl:$(IDL_DIR)/external/idl.export" ;; \
"HP-UX") make idl_tools \
"SO_EXT=sl" \
"X_CFLAGS=+z -Aa -D_HPUX_SOURCE" \
"X_LD_FLAGS= -b -Brestricted +s" ;; \
"IRIX") make idl_tools \
"X_LD_FLAGS=-Bsymbolic -shared -all -rdata_shared" ;; \
"IRIX64") make idl_tools \
"X_LD_FLAGS=-Bsymbolic -shared -all -rdata_shared" ;; \
"OSF1") make idl_tools \
"X_CFLAGS=-float" \
```

```

"X_LD_FLAGS=-expect_unresolved '*' -shared -all" ;;\
"Linux" ) make idl_tools \
"X_CFLAGS= -fPIC" \
"X_LD_FLAGS= -shared -Bsymbolic --warn-once -noinherit-exec " ;; \
*) echo "This system is not supported" ;; \
    esac \
fi

# The following is the entry point for the recursive build

idl_tools : idl_tools.$(SO_EXT)
@date

idl_tools.$(SO_EXT) : idl_tools.o idl_sock.o idl_snd.o idl_gzip.o
idl_mesh.o \
idl_twain.o idl_defproc.o idl_sharr.o zlib/libz.a
-$(LD) $(X_LD_FLAGS) -o idl_tools.$(SO_EXT) idl_tools.o idl_sock.o \
idl_snd.o idl_gzip.o idl_mesh.o idl_twain.o idl_defproc.o \
idl_sharr.o -Lzlib -lz

zlib/libz.a :
cd zlib ; ./configure ; make

clean :
cd zlib; make clean
rm -f idl_tools.o idl_tools.so idl_tools.sl idl_tools.a *.o \
so_locations

FILES = idl_tools.c idl_sock.c idl_tools.dlm idl_tools.export
idl_tools.h \
idl_tools.def idl_snd.c idl_gzip.c idl_mesh.c idl_twain.c \
twain.h table.c table.h idl_twain_glue.c idl_twain_glue.h \
idl_defproc.c dynacall.c dynacall.h idl_sharr.c \
idl_tools.dsw twndebug.h COPYING.txt README.txt idl_tools.dsp
Makefile

tar : $(FILES)
tar czvf idl_tools.tar.gz $(FILES)

```
