Lightspeed Blackbox Developers Kit (BDK)

How-To Compile the Linux Blackbox Trading System

Version 1.03
September 30, 2010

Compiling a Blackbox Trading System

The following files are required to compile a Blackbox Trading System using Lighstpeed's Blackbox Deveolpers Kit (BDK).

File	Description
liblssdk.a	This file is the BDK Library. Compiled from of the BDK components common to all Blackbox Trading Systems.
customer_sample.c	This file contains software written by the customer. It will typically contain the customer's Trading Strategy Logic and Position Management Logic. This file contains stub functions for all the BDK defined call-back functions.
customer_sample.h	This file is the customer's header file.
sdk_proto.h	This file contains prototypes for all the BDK defined function calls. A function call refers to a function provided by the BDK Library that can be called by the customer's software.
platform.h	This file contains only one line of code. It contains a #define that specifies whether the Blackbox is to be built as a Linux application or a Windows application.
makefile	"make" is a utility for building the executable program from libraries and source code. This file (makefile) defines the set of rules that will be used to perform the build.

System Requirements

Building the Blackbox Trading System should be performed on a Linux system with a kernel version of 2.4 or later. The GNU Compiler Collection (GCC) must be installed on the Linux system.

Compilation Procedure

1. Create a directory on the Linux system.
2. Obtain the six files described above from Lightspeed and copy them to the directory created in step 1.
3. Edit the file makefile and chose a name for the executable program. The sixth line of the make file contains PNAME=bot. bot is the default name. Change bot to the desired program name.
4. Type make on the command line. This will compile and link the executable program.

Makefile

```
#
#       Makefile for Blackbox Trading System
#

# Program Name
PNAME = bot

# Compiler
CC=gcc

# Compiler Flags
#CFLAGS = -O3 -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE
CFLAGS = -g -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE

LIBS = -L. -llssdk

INCLUDE = -I .

OBJS = customer_sample.o

$(PNAME): $(OBJS)
      $(CC) $(CFLAGS) $(INCLUDE) -o $(PNAME) $(OBJS) $(LIBS)

customer_sample.o: customer_sample.c
       $(CC) $(CFLAGS) $(INCLUDE) -c customer_sample.c

clean:
      - rm -f *.o $(PNAME)
```