6 Using the HLA Command-Line Compiler

Once you've installed HLA and verified that it is operational, you can run the HLA compiler. The HLA compiler consists of two executables: hla(.exe)[1], which is a shell that processes command line arguments, compiles ".hla" files to ".asm" files, assembles the ".asm" files by calling an assembler, and links the resulting files together using a linker program; the second executable is hlaparse(.exe) which compiles a single ".hla" file to an assembly language file. Generally, you would only run hla(.exe). The hla(.exe) program automatically runs the hlaparse(.exe) and assembler/linker programs. The hla(.exe) command uses the following syntax:

```
hla  optional_command_line_parameters  Filename_list
```

The filenames list consists of one or more unambiguous filenames having the extension: ".hla", ".asm" or ".obj"/".o"[2]. HLA will first run the hlaparse(.exe) program on all files with the HLA extension (producing files with the same basename and an ASM extension). Then HLA runs the assembler on all files with the ".asm" extension (including the files produced by hlaparse). Finally, HLA runs the linker to combine all the object files together (including the ".obj"/".o" files the assembler produces). The ultimate result, assuming there were no errors along the way, is an executable file (with an EXE extension under Windows, with no extension under Linux/FreeBSD/ Mac OSX).

HLA supports the following command line parameters:

```
options:
  -@            Do not generate linker response file.
  -@@           Always generate a linker response file.
  -thread       Enable thread-safe code generation and linkage.
  -axxxxxx      Pass xxxxx as command line parameter to assembler.
  -dxx          Define VAL symbol xx to have type BOOLEAN and value TRUE.
  -dxx=yy       Define VAL symbol xx to have type STRING and value "yy".
  -e:name       Executable output filename (appends ".exe" under Windows).
  -x:name       Executable output filename (does not append ".exe").
  -b:name       Binary object file output name (only when using HLABE).
  -i:path       Specifies path to HLA include file directory.
  -lib:path     Specifies path to the HLALIB.LIB file.
  -license      Displays copyright and license info for the HLA system.
  -lxxxxx       Pass xxxxx as command line parameter to linker.
  -m            Create a map file during link
  -p:path       Specifies path to hold temporary working files.
  -r:name       <name> is a text file containing cmd line options.
  -obj:path     Specifies path to place object files.
  -main:name    Use 'name' as the name of the HLA main program.
  -source       Compile to human readable source file format.
  -s            Compile to .ASM files only.
  -c            Compile and assemble to object files only.
  -fasm         Use FASM as back-end  (applies to -s and -c)
  -gas          Use Gas as back-end  (Linux/BSD, applies to -s and -c)
  -gasx         Use Gas as back-end  (Mac OSX, applies to -s and -c)
```

1. The ".exe" suffix appears only in the Windows' version.
2. Windows object files use the ".obj" suffix while Linux object files have the ".o" suffix. Although Linux users who write assembly code with Gas typically use a ".s" or ".S" suffix, HLA still uses ".asm" since Gas happily accepts this.

```
   -hla         Produce a pseudo-HLA source file as output (implies -s).
   -hlabe       (Default) Produce obj file using the HLA Back Engine.
   -masm        Use MASM as back-end assembler (applies to -s and -c)
   -nasm        Use NASM as back-end assembler (applies to -s and -c)
   -tasm        Use TASM as back-end assembler (applies to -s and -c)
   -sym         Dump symbol table after compile.
   -win32       Generate code for Win32 OS.
   -linux       Generate code for Linux OS.
   -freebsd     Generate code for FreeBSD OS.
   -macos       Generate code for Mac OSX.
   -test        Send diagnostic info to stdout rather than stderr (This
                 option is intended for HLA test/debug purposes).
   -v           Verbose compile.
   -level=h     High-level assembly language
   -level=m     Medium-level assembly language
   -level=l     Low-level assembly language
   -level=v     Machine-level assembly language (very low level).
   -w           Compile as windows app (default is console app).
   -?           Display this help message.
```

Note that HLA ignores case when processing command line parameters (unlike typical Linux/FreeBSD/Mac OSX programs). For example, "-s" is equivalent to "-S" when specifying a command line parameter.

-@
-@@

HLA will produce a "linker response file" that it supplies to the linker program during the link phase. This linker response file contains necessary segment declarations and other vital linker information. By default, HLA uses any existing "*basename*.link" file (where *basename* is the base name of the file you are compiling) whenever you run the compiler; it will create a new "*basename*.link" file only if one does not already exist. The "-@" option tells HLA not to create a new ".link" file, even if one does not already exist. The "-@@" option tells HLA to always create a ".link" file, even if one already exists.

If you specify multiple "*basename*.hla" filenames on the command line, HLA only generates a single "*basename*.link" file using the name of the first "*basename*.HLA" file it encounters.

-r:filename

The "-r:filename" option lets you specify a response file containing a sequence of HLA command-line parameters. The file specified after this option must contain a sequence of HLA command-line parameters, one per line, which HLA executes exactly as though they were specified on the command line. E.g.,

```
sampleFile.resp:

-fasm
sampleFile.hla

The following command treats each of the above lines as separate HLA
command-line parameters:

hla -r:sampleFile.resp
```

-aXXXXX

The -a*XXXXX* option lets you pass assembler-specific command line options to the assembler during the assembler phase. This option is ignored if you use the -s option or you're compiling directly to object code using the HLA back engine. One common form of this command often used with the MASM assembler is "-aZi -aZf" that tells MASM to generate debugging information in the object file (for use with the Visual Studio debugger program).

-c

The -c option tells HLA to run the hlaparse compiler and the (default) assembler, producing "*basename*.obj"/"*basename*.o" files. HLA will process all filenames on the command line that have ".hla" or ".asm" extension, but it will ignore any filenames with ".obj" or ".o" extensions. If you compile an HLA unit without compiling an HLA program at the same time, you will need to use this option or the linker will complain about not finding the main program.

One common use of this option is to compile HLA units to object files. Since HLA units do not contain a main program, you cannot compile an HLA unit directly to an executable. To compile an HLA unit separately (i.e., without compiling an HLA main program during the same HLA.EXE invocation) you must specify the "-c" option or the compilation will generate an error when it attempts to link the program.

A second reason for using the "-c" option is that you want to explicitly run the linker yourself and supply linker command line options that are different from those that HLA automatically provides.

-fasm

Tells HLA to use FASM as the back-end assembler. This command-line overrides any back-end assembler specification previously on the command-line. Object code and executable file output with this command are available only under Windows. Source output is available under any operating system.

-gas

Tells HLA to use GAS as the back-end assembler. This command-line overrides any back-end assembler specification previously on the command-line. Object code and executable file output with this command are available only under Linux and FreeBSD. Source output is available under any operating system.

-gasx

Tells HLA to use GAS as the back-end assembler. This command-line overrides any back-end assembler specification previously on the command-line. Object code and executable file output with this command are available only under Mac OS X. Source output is available under any operating system.

-hla

Tells HLA to produce a human-readable pseudo-HLA-syntax assembly language output file. This command-line option implies "-source" and "-s". The main use for this option is to see how HLA expands macros, high-level-language-like statements, and other code.

-hlabe

Tells HLA to use the HLA Back Engine as the back-end assembler. This command-line overrides any back-end assembler specification previously on the command-line. This command causes HLA to directly produce an object code file without using an external back-end assembler. If you specify both of the "-source" and "-s" command-line options along with "-hlabe", then HLA will produce a human-readable ".asm" file rather than an object file; this option is useful for debugging HLA or for those curious about how HLABE operates.

-masm

Tells HLA to use MASM as the back-end assembler. This command-line overrides any back-end assembler specification previously on the command-line. Object code and executable file output with this command are available only under Windows. Source output is available under any operating system.

-nasm

Tells HLA to use NASM as the back-end assembler. This command-line overrides any back-end assembler specification previously on the command-line. Object code and executable file output with this command are available only under Windows. Source output is available under any operating system.

-tasm

Tells HLA to use TASM as the back-end assembler. This command-line overrides any back-end assembler specification previously on the command-line. Object code and executable file output with this command are available only under Windows. Source output is available under any operating system. Note that TASM support in HLA is deprecated, so it's not a good idea to depend on this option.

-d:XXXXX{=YYYYY}

The -d*XXXXX* option tells HLA to define the symbol *XXXXX* as a boolean **val** constant and initialize it with the value **true**. Generally you use such symbols to control the emission of code during assembly using statements like "#if(@defined(*XXXXX*)) ..."

The -d*XXXX*=*YYYY* option tells HLA to define the symbol *XXXX* as a string **val** constant and give it the initial value "*YYYY*".

-b:name

When compiling to an object file using the HLA back-engine (rather than a back-end assembler), this option specifies the name of the binary object file. By default, HLA uses the base name (before the ".hla" suffix) with a ".obj" (windows) or ".o" (*NIX) suffix. With the "-b:name" option you may specify a different name. Note that if you do not supply an ".obj" or ".o" suffix at the end of "-b:name" because HLA will automatically attach the suffix.

-e:name

By default, HLA creates an executable filename using the extension ".exe" (Windows) or without an extension (*NIX) and the *basename* of the first filename on the command line. You can use the -e *name* option to specify a different executable file name.

-x:name

Similar to the -e:name option, except there is no automatic ".exe" suffix applied under Windows. This lets you explicitly supply the suffix (e.g., ".dll" under Windows, or to force a ".exe" under *NIX).

-lXXXXX

The -l*XXXXX* option passes the text *XXXXX* on to the linker as a command line option. One common command to pass to the Microsoft linker is "-lDEBUG" that tells the linker to generate debugging information in the object file.

-m

The -m option tells the Microsoft linker or POLINK to produce a map file during the link phase. This is equivalent to the "-lmap" option. The *NIX version of HLA ignores this option.

-s

The -s option tells the HLA program to run only the hlaparse compiler to produce an assembly language source file; HLA will not run a back-end assembler or linker. As a result, HLA ignores any ".asm" or ".obj" filenames you supply on the command line. This option is useful if you wish to view the output of an HLA compilation in some other assembler's source format without producing any actual object code. This option must be used with the "-source" command-line option if you are using the HLA back engine and you wish to produce a human-readable source file of the HLABE output.

-sym

The -sym option dumps the symbol table after compiling each file with an HLA extension. This option is primarily intended for testing and debugging the HLA compiler; however, this information can be useful to the HLA programmer on occasion.

-thread

The -thread option tells HLA to generate thread-safe code (for the code it emits) and link in the thread-safe version of the HLA standard library. Specifying this command-line option sets the HLA @thread object to true, which you can test during the compilation of an HLA source file (that must behave differently for thread-safe versus non-thread-safe code).

-test

The -test option is intended for hlaparse testing and debugging purposes only. It causes the compiler to send all error messages to the standard output device rather than the standard error device. This allows the test code to redirect all errors to a text file for comparison against other files. This command also causes HLAPARSE to emit some extra comment information to the assembly language output file when producing output files for one of the back-end assemblers (other than the HLA back engine).

-v

The -v option (verbose) causes HLA to print additional information during compile to show the progress of the compilation. This option also prints certain statistics, such as the number of lines per second that HLA compiles.

-w

The -w option informs HLA that you are compiling a standard Windows (GUI) application rather than a console application. By default, HLA assumes that you are compiling a executable that will run from the command window. If you want to write a full Windows application, you will need to supply this option to tell HLA not to link the code for console operation. Obviously, this option doesn’t apply to *NIX systems. The "-w" option tells HLA to invoke the linker using the command line option

-subsystem:windows

rather than the default

-subsystem:console

This provides a convenient mechanism for those who wish to create win32 GUI applications. Most likely, however, if you wish to create GUI applications, you will run the linker explicitly yourself (as this document will explain), so you’ll probably not use the "-w" option very frequently. It’s great for some short GUI demos, but larger GUI programs will probably not use this option. This option is only active if HLA compiles the program to an executable. If you compile the program to an OBJ or ASM file, HLA ignores this option.

If you want to develop Win32 GUI apps, look at Randy Hyde’s book "Windows Programming in Assembly". This book provides the linker commands and makefiles for generation such applications (as well as describing how you actually write such code).

-p:path

During compilation, HLA produces several temporary files (that it doesn't delete, because they may be of interest to the HLA user). These files have a habit of cluttering up the current working directory. If you prefer, you can tell HLA to place these files in a temporary directory so they don't clutter up your working directory. One way to accomplish this is by using the "-p:*dirpath*" command line option. For example, the option "-p:c:\hla\tmp" tells HLA to put all temporary files (for the current assembly) into the "c:\hla\tmp" subdirectory (which must exist). Note that you can set also set the temporary directory using the hla "hlatemp" environment variable. The "-p:*dirpath*" option will override the environment variable (if it exists). See the description of the hlatemp environment variable for more details.

-obj:path

During compilation, HLA normally writes all object files to the current working directory. Some programmers have requested a way to specify a different directory for the .OBJ (.o under *NIX) files that HLA produces. You can accomplish this using the "-obj:*dirpath*" command line option. The *dirpath* item has to be the path to a valid directory. HLA places all object files produced by the compiler and/or resource editor in this directory. Note that, unlike the -p option, there is no environment variable that lets you permanently set this path. You must specify the path on a compilation-by-compilation basis (use a makefile if you get tired of typing the path in on each compilation).

-level=h

-level=m

-level=l

-level=v

The -level options enable or disable certain HLA language features. These command-line options are intended for use in programming courses where the instructor needs to batch compile dozens or even hundreds of student projects at one time. This allows the instructor to ensure that the students aren't using high-level control constructs that are inappropriate for that point in the course. For example, towards the end of a course, most instructors don't allow the use of various high-level control constructs; some instructors may never allow them. The "-level" command-line options will "turn off" various statements in the HLA language so that the HLA compiler will report an error if the student attempts to use them in a source file.

The default, "-level=h" (high) enables the entire HLA language.

The "-level=m" (medium level) disables high-level language control constructs, such as "if", "while", and "for" but still allows the use of high-level-like procedure calls in the HLA language. Medium-level assembly language also allows the use of exceptions using HLA's try..except..endtry and raise statements.

The "-level=l" (low-level assembly) disables all high-level control constructs other than the exception-handling statements and disables high-level-like procedure calls in HLA. This option also disables automatic stack frame generation and clean up in HLA procedures (that is, the programmer will be responsible for writing that code themselves).

The "-level=v" (very low-level assembly) option disables all high-level control constructs including exception handling. Only machine instructions (and user written macros) are legal in the source file. No high-level control constructs or high-level procedure calls are allowed.

-?

The -? option cause HLA to dump the list of command line options and immediately quit without further work.

Note that the command line options this document describes are for HLA v2.2 and later only. Earlier versions of HLA used a different command line set. See the documentation for the specific version you’re using if you have questions.

-license

This command displays license information for the entire HLA system. Although the HLA source code written by Randall Hyde is all public domain, certain components of the HLA system, including the back-end assemblers, the linker, and the resource editor, may come from other sources. The "-license" command-line parameter lists license information about these other products.