|  |  |
| --- | --- |
|  |  |
|  | Intel IMP Compiler User Guide  Introduction  The Intel IMP compiler is supported on 16 bit MS-DOS, 32 bit MS-Windows and 32 bit UNIX systems. The compiler produces standard object files for each of these systems that are then linked with the appropriate run-time libraries to produce executable programs. The object files produced are compatible with existing linkers, loaders and debuggers on these systems, and subject to certain restrictions, can also be linked with existing libraries in other languages, particularly C.  You should note that the IMP compiler package can only be used if you already have some basic development tools on your machine - you need a linker and a C library. The development tools used for developing the compiler itself were:   * 16 bit MS-DOS using Microsoft Visual C version 1.52 and Codeview. * 32 bit MS-Windows using Microsoft Visual Studio version 6.0. * 32 bit UNIX using Gnu tools.   The 16 bit MS-DOS linker is widely available (a version of the linker was supplied free with MS-DOS) and a suitable version is included in our [download area](https://web.archive.org/web/20050407095639/http:/imp.nb-info.co.uk/download.htm). The Gnu tools for UNIX are also widely available. Unfortunately the 32 bit Windows tools are not freely distributable, and a Microsoft development kit must be installed to build 32 bit executables.  Installing the IMP compiler  The compiler package consists of the three executable passes of the compiler, a shell script that runs them, and the IMP library. You should unzip and copy the files to the appropriate location on your system.  DOS and Windows users should put the compiler and the library in a simple directory (such as C:\IMP) and then add that directory to their PATH and LIB environment variables:  set PATH=C:\IMP;%PATH% set LIB=C:\IMP;%LIB%  The DOS and Win-32 versions of the compiler use different file names, so it is possible to install both on the same machine. Note however that they cannot both be used in the same session because both depend on the appropriate linker being on the search path, and the Microsoft 16 bit linker and the Microsoft Win-32 linker are both called LINK.EXE.  UNIX users should install the binaries and libraries according to their own local policy. Note that this means you should inspect/edit the "install" section of the Makefile before running it - see the README for more information.  Using the IMP compiler  All three versions of the compiler work the same way, with a shell script wrapper:  Syntax  **imp77** [**-c**] [**-Fs**] [**-Fc**] *filename*  Description  The imp77 command invokes the Imp-77 compiler on a single Imp source file. For the MS-DOS and Windows versions of the command, the extension .IMP is always added to the name and therefore the extension should not be given on the command line. The UNIX version expects you to provide the extension in the command, and accepts either .imp or .i. Unless the **-c** flag is given, the resulting object file is passed to the link/loader and an executable file is produced.  Options   |  |  | | --- | --- | | **-c** | Suppresses the loading phase of the compilation and forces an object file to be produced. | | **-Fs** | Produces a source listing file. The listing file has the base name of the source file plus the extension .LST. | | **-Fc** | Produces a combined source, assembly and machine-code listing. The listing file has the base name of the source file plus the extension .COD. |   Run-time Library Support  The library supplied with the compiler includes all of the standard IMP library routines and a few of the common extensions. If there are any missing that you would like to contribute, please do so!  There are two run-time environments available for your programs. The default is that your program is assumed to be a "classic" IMP program, using **begin** and **end of program**. When the program runs, the command line is treated as a set of file parameters. For DOS users the inputs are separated from the outputs with a "/" character with no spaces between them - formally the syntax is:  **myprogram** [*infile1*[*,infile2*[*,infile3*]]][/*outfile1*[*,outfile2*[*,outfile3*]]]  For UNIX users the inputs are separated from the outputs with white space, so the syntax is:  **myprogram** [*infile1*[*,infile2*[*,infile3*]]] [*outfile1*[*,outfile2*[*,outfile3*]]]  If you want to have additional control over the command line parameters, an alternative library is provided which assumes that your program is of the form:  **external** **routine** MyProg **alias** "\_\_impmain" (**integer** argc, **string**(255)**name array name** argv)  The semantics of this contruction is based on the C equivalent, where argv is declared as argv(0:argc-1) and argv(0) will usually be the program name. To use this form you must link your program with the alternate IMP library. This involves compiling the program without linking by using the **-c** switch, and then linking manually with the alternate library. For example in UNIX this would be:  imp77 -c myprog.imp gcc -o myprog myprog.o -li77arg  See Also  For further information on IMP-77 programming, the classic reference text is available [here](https://web.archive.org/web/20050407095639/http:/imp.nb-info.co.uk/imp77.pdf) (IMP77.PDF, 130kbytes). |