WIKIPEDIA

Program database

Program database (**PDB**) is a file format (developed by Microsoft) for storing debugging information about a program (or, commonly, program modules such as a DLL or EXE). PDB files commonly have a .pdb extension. A PDB file is typically created from source files during compilation. It stores a list of all symbols in a module with their addresses and possibly the name of the file and the line on which the symbol was declared. This symbol information is not stored in the module itself, because it takes up a lot of space.

Program database

Filename extension	`.pdb`
Internet media type	application/octet-stream
Developed by	Microsoft
Type of format	Debug

Contents

Applications

When a program is debugged, the debugger loads debugging information from the PDB file and uses it to locate symbols or relate current execution state of a program source code. Microsoft Visual Studio uses PDB files as its primary file format for debugging information.

Another use of PDB files is in services that collect crash data from users and relate it to the specific parts of the source code that cause (or are involved in) the crash.

Microsoft compilers will, under appropriate options, store information in a single PDB about types found in the compiled sources. Debug information specific to each source is stored in the compiled object file, and contains references to types in the PDB. Each compilation will add to the PDB any types that are not already found there, so that references in already compiled object files remain valid.

The Microsoft linker, under appropriate options, builds a complete new PDB which combines the debug information found in its input modules, the types referenced by those modules, and other information generated by the linker. If the link is performed incrementally, an existing PDB is modified by adding replacing only the information pertaining to added or replaced modules, and adding any new types not already in the PDB.

PDB files are usually removed from the programs' distribution package. They are used by developers during debugging to save time and gain insight.

Extracting information

The PDB format is documented here (https://github.com/Microsoft/microsoft-pdb), information can be extracted from a PDB file using the DIA (Debug Interface Access) interfaces, available on Microsoft Windows. There are also third-party tools that can also extract information from PDB such as radare2 and pdbparse (https://github.com/moyix/pdbparse)

Multiple stream format

The PDB is a single file which is logically composed of several sub-files, called **streams**. It is designed to optimize the process of making changes to the PDB, as performed by compiles and incremental links. Streams can be removed, added, or replaced without rewriting any other streams, and the changes to the metadata which describes the streams is minimized as well.

The PDB is organized in fixed-size **pages**, typically 1K, 2K, or 4K, numbered consecutively starting at 0.

Note: It is presumed that all numeric information (*e.g.,* stream and page numbers) is stored in little-endian form, the native form for Intel x86 based processors. The pdbparse Python code makes this assumption.

Stream

Each stream in the PDB occupies several pages, which aren't necessarily consecutively numbered. The stream has a number and a length. The stream content is the concatenation of its pages, truncated to the stream's length.

Metadata format

The function of the PDB metadata is to identify all of the component streams, giving the length, and sequence of pages for each stream. Streams are numbered consecutively starting with 0. There is also a root stream, unnumbered, which contains some of the metadata.

Header

The PDB begins with a header, consisting of:

- Signature, used to identify and validate the specific format. The length of the signature varies with the specific format.
- The remainder of the header varies with the format identified by the signature.

The header may be longer than a single page.

Microsoft tools use two PDB formats:

Version 2

Signature is "`Microsoft C/C++ program database 2.00\r\n\032JG\0\0`"(44 bytes).

Remainder of the header consists of:

- Page size, 4 bytes.
- Start page, 2 bytes.
- Number of file pages, 2 bytes.
- Root stream size, 4 bytes.
- reserved, 4 bytes.
- Root stream page number list, 2 bytes per page, enough to cover the above Root stream size.

Version 7

Signature is "`Microsoft C/C++ MSF 7.00\r\n\x1ADS\0\0\0`"(32 bytes).

Remainder of the header consists of:

- Page size, 4 bytes.
- Allocation table pointer, 4 bytes. The meaning of this is unknown. There appears to be an allocation table, an array of 65,536 bits (8,192 bytes), located at the end of the PDB, and a 1-bit means a page that is not being used.
- Number of file pages, 4 bytes.
- Root stream size, 4 bytes.
- reserved, 4 bytes.
- Page number of the Root stream page number list. It does not indicate the location of the Root stream itself, only of the page containing the structure which points to its pages. At that page, the Root stream page number list indicates the pages where the Root stream is stored. It contains 4 bytes per page, enough to cover the above Root stream size.

Root stream

The root stream describes all of the PDB streams starting with stream 0. Its contents vary with the PDB format version.

Version 2

The root stream consists of:

- Number of streams, 2 bytes.
- Reserved, 2 bytes.
- For each stream:
  - Stream size, 4 bytes.
  - Reserved, 4 bytes.

- For each stream:
  - Stream page number list, 2 bytes per page, enough to cover above stream size.

Version 7

The root stream consists of:

- Number of streams, 4 bytes.
- For each stream:
  - Stream size, 4 bytes.
- For each stream:
  - Stream page number list, 4 bytes per page, enough to cover above stream size.

Stream contents

Microsoft tools store different sorts of information in different numbered streams. Some stream numbers have a fixed information type associated with them, and other streams are identified in the aforementioned fixed type streams.

Stream 1 is used to verify that the PDB is the same file referred to in an executable or object file stream.

- Version, 4 bytes.
- Time date stamp, 4 bytes.
- Age, 4 bytes. This is the number of times this PDB has been modified since its creation.
- GUID, 16 bytes.
- Total length of following names, 4 bytes. Followed by null-terminated character strings.

Stream 2 and **stream 4** hold types information. Actual type records define types used in the program. The structure of these records can be found in the file cvinfo.h provided by Microsoft. There are two flavors of records, each with its own set of index numbers: type IDs and types; only types are stored in stream 2 and only type IDs are stored in stream 4. The indices are used to refer to these records from within symbol records and other type records.

- A header:
  - Version, 4 bytes.
  - Header size, 4 bytes.
  - Minimum and maximum (last + 1) index for type records (4 bytes each).
  - Size of following data, 4 bytes, to the end of the stream.
- Hash information:
  - Stream number, 2 bytes with 2 bytes padding.
  - Hash key, 4 bytes.
  - Buckets, 4 bytes.
  - HashVals, TiOff, and HashAdj, each composed of an offset and length, each 4 bytes.
- Type records, variable length, count = (maximum - minimum) from above header.

Stream 3 is a directory for other streams. Note, it is not present in Version 2, nor in a PDB produced by a compiler. The stream starts with a header which is padded to be 64 bytes in total

PDB Stream 3 Header (struct NewDBIHdr)[1] (https://github.com/Microsoft/microsoft-pdb/blob/master/PDB/dbi/dbi.h)

Offset	Size	Name	Description
0	4	Signature	Header identifier, == 0xFFFFFFFF
4	4	HeaderVersion	Version of the Header
8	4	Age	
12	2	snGSSyms	

14	2	usVerAll	`union {`
			`    struct {`
			`        USHORT      usVerPdbDllMin : 8; // minor version and`
			`        USHORT      usVerPdbDllMaj : 7; // major version and`
			`        USHORT      fNewVerFmt     : 1; // flag telling us we have rbld stored elsewhere (high bit of original major version)`
			`    } vernew;                           // that built this pdb last.`
			`    struct {`
			`        USHORT      usVerPdbDllRbld: 4;`
			`        USHORT      usVerPdbDllMin : 7;`
			`        USHORT      usVerPdbDllMaj : 5;`
			`    } verold;`
			`    USHORT          usVerAll;`
			`};`

16	2	snPSSyms	
18	2	usVerPdbDllBuild	build version of the pdb dll that built this pdb last
20	2	snSymRecs	
22	2	VerPdbDllRBld	rbld version of the pdb dll that built this pdb last
24	4	cbGpModi	size of rgmodi substream
28	4	cbSC	size of Section Contribution substream
32	4	cbSecMap	size of section map
36	4	cbFileInfo	size of file info stream
40	4	cbTSMap	size of the Type Server Map substream
44	4	iMFC	MFC Index
48	4	cbDbgHdr	size of optional DbgHdr info appended to the end of the stream
52	4	cbECInfo	number of bytes in EC substream, or 0 if no EC enabled Mods

56	2	flags	`struct _flags {`
			`    USHORT  fIncLink:1;     // true if linked incrmentally (really just if ilink thunks are present)`
			`    USHORT  fStripped:1;    // true if PDB::CopyTo stripped the private data out`
			`    USHORT  fCTypes:1;      // true if this PDB is using CTypes.`
			`    USHORT  unused:13;      // reserved, must be 0.`
			`} flags;`

58	2	wMachine	Machine identifier, same as used in COFF object format, *e.g.,* hex 8664 for Intel x86 64-bit
60	4	RESERVED	future expansion, pad to 64 bytes

- Module information, variable length. Total size in above header. There is one of these for each object module used by the linker
  - Opened, 4 bytes.
  - Symbol info.
    - Section number, 2 bytes + 2 bytes padding.
    - Offset and size, 4 bytes each.
    - Flags, 4 bytes.
    - Module number, 2 bytes + 2 bytes padding.
    - CRCs for section data and relocations data, 4 bytes each.
  - Flags, 2 bytes.
  - Stream number, 2 bytes.
  - Symbols size, 4 bytes.
  - Old and new line number info sizes, 4 bytes each.
  - Number of source files, 2 bytes + 2 bytes padding.
  - Offsets, 4 bytes.
  - niSource and niCompiler, 4 bytes each.
  - Module name, null terminated byte string.
  - Object name, null terminated byte string.
  - Padding to multiple of 4 bytes.
- Section contributions, section headers, file info, ts map, and EC info. Their sizes are found in the above header.
- Debug header,
  - Stream numbers for Old Frame Pointer Omission, Exceptions, Fixups, Object Maps to and from Source, Section Headers, Token Ring IDs, Xdata, Pdata, New Frame Pointer Omission, and Section Header Origin. 2 bytes each.

See also

- Debug symbol

External links

- The PDB format is documented here Information from Microsoft about the PDB format. (https://github.com/Microsoft/microsoft-pdb)
- Microsoft MSDN documentation on DIA (http://msdn.microsoft.com/en-us/library/t6tay6cz.aspx)
- How To Inspect the Content of a Program Database (PDB) File (http://www.codeproject.com/KB/bugs/PdbParser.aspx)
- Symbols and Symbol Files, MSDN (http://msdn.microsoft.com/en-us/library/ff558825(v=vs.85).aspx)
- What’s inside a PDB File? / Visual C++ Team Blog (https://blogs.msdn.microsoft.com/vcblog/2016/02/08/whats-inside-a-pdb-file/)
- PDB structure according to LLVM (https://llvm.org/docs/PDB/index.html)