NAME

snacc − ASN.1 to C, C++ or type table Compiler

SYNOPSIS

```
snacc [−h] [−P] [−t] [−e] [−d] [−p] [−f]
    [−c | −C | −idl | −T <table output file>]
    [−u <useful types ASN.1 file>]
    [−mf <max file name len>]
    [−l <neg number>]
    [−meta] [−tcl <module.type>]
    [−novolat]
    <ASN.1 file list>
```

For complete and current documentation, refer to the snacc manual.

DESCRIPTION

Snacc (Sample Neufeld Asn.1 to C/C++ Compiler) generates C or C++ source code for BER encode and decode routines as well as print and free routines for each type in the given ASN.1 modules. Alternatively, snacc can produce type tables that can be used for table based/interpreted encoding and decoding. The type table based methods tend to be slower than their C or C++ counterparts but they usually use less memory (table size vs. C/C++ object code).

Most of the 1990 ASN.1 features are parsed although some do not affect the generated code. Fairly rigourous error checking is performed on the ASN.1 source; any errors detected will be reported (printed to stderr).

Each file in the ASN.1 file list should contain a complete ASN.1 module. ASN.1 modules that use the IMPORTS feature must be compiled together (specify all necessary modules in the ASN.1 file list). The generated source files will include each module's header file in the command line order. This makes it important to order the modules from least dependent to most dependent on the command line to avoid type ordering problems. Currently, snacc assumes that each ASN.1 file given on the command line depends on all of the others on the command line. No attempt is made to only include the header files from modules referenced in the import list for that module.

If the target language is C, snacc will generate a **.h** and **.c** file for each specified ASN.1 module. If the target language is C++, snacc will generate a **.h** and **.C** file for each module. The generated file names will be derived from the module names.

OPTIONS

−h Help. Prints a synopsis of snacc and exits.

−c Generate C source code. This is the default behaviour of snacc. Only one of **−c** , **−C** or **−T** should be specified.

−C Generate C++ source code.

−novolat
Generate ‘‘return *this’’ after calling ‘‘abort()’’. (Some broken compilers don't know about volatile functions, or their abort() isn't correctly typed.)

-meta Generate meta code that describes the generated types. Implies −C.

-tcl *module.type*[,*module.type*] Generate code for a Tcl interpreter where *module.type* are the top level PDUs. Implies −meta.

−T *file*This causes snacc to generate type tables and write them to the given file.

−P This causes snacc to print the parsed ASN.1 modules to stdout after the types have been linked, sorted, and processed. This option is useful for debugging snacc and observing the modifications snacc performs on the types to make code generation simpler.

−t Generate type definitions in the target language for each ASN.1 type.

−v Generate value definitions in the target language for each ASN.1 value. Currently value definitions are limited to INTEGERs, BOOLEANs and OBJECT IDENTIFIERs.

−e Generate encode routines in the target language for each ASN.1 type.

−d Generate decode routines in the target language for each ASN.1 type.

−p Generate print routines in the target language for each ASN.1 type.

−f Generate free routines in the target language for each ASN.1 type. This option only works when the target language is C.

If none of the **−t, −v, −e, −d, −p, or −f** options are given on the command line, snacc assumes that all of them are in effect. They do not affect type table generation.

−u *file*Read the useful types definitions from the ASN.1 module in file *file for linking purposes. For some ASN.1 specifications, such as SNMP,*
the useful types are not needed. The types in the given useful types file are globally available to all modules; a useful type definition is overridden by a local or explicitly imported type with the same name. The current list of useful types is:

ObjectDecscriptor
NumericString
PrintableString
TeletexString
T61String
VideoTexString
IA5String
GraphicString
ISO646String
GeneralString
UTCTime
GeneralizedTime
EXTERNAL

−mf *number*This causes the generated source files to have a
maximum length of *number characters, including their suffix. The number must be at least 3. This option is useful for supporting operating* systems that only support short file names. A better solution is to shorten the module name of each ASN.1 module.

−l *number*This is fairly obscure but may be useful. Each error that the
decoders can report is given an id number. The number*number is wher e the error ids start decreasing from as they are assigned to* errors . The default is −100 if this option is not given. Avoid using a number in the range −100 to 0 since they may conflict with the library routines' error ids. If you are re-compiling the useful types for the library use −50. Another use of this option is to integrate newly generated code with older code; if done correctly, the error ids will not conflict.

FILES

snacc/asn1specs/asn-useful.asn1	ASN.1 useful types module (use with −u option)
snacc/c-lib/inc/	C runtime library include files
snacc/c-lib/libasn1csbuf.a	C SBuf runtime library
snacc/c-lib/libasn1cmbuf.a	C MinBuf runtime library

snacc/c-lib/libasn1cebuf.a	C ExpBuf runtime library
snacc/c++-lib/inc/	C++ runtime library include files
snacc/c++-lib/libasn1c++.a	C++ runtime library
snacc/c-lib/inc/tbl*/	Type table runtime library include files
snacc/c-lib/libasn1ctbl.a	Type table runtime library
snacc/tbl-tools/	Source code for table based tools (mkchdr, ptbl, pval)
snacc/c-examples/	directory with ASN.1 to C examples
snacc/c++-examples/	directory with ASN.1 to C++ examples
snacc/tbl-example	directory with an ASN.1 to type table example
snacc/doc	directory with snacc documentation and this man page

BUGS

Snacc has problems with the following case:

```
Foo ::= SEQUENCE
{
   id IdType,
   val ANY DEFINED BY id
}

IdType ::= CHOICE
{
   a INTEGER,
   b OBJECT IDENTIFIER
}
```

The error checking pass will print an error to the effect that the id type must be INTEGER or OBJECT IDENTIFER. To fix this you must modify the error checking pass as well as the code generation pass. To be cheap about it, disable/fix the error checking and hand modify the generated code.

The hashing code used for handling ANY DEFINED BY id to type mappings will encounter problems if the hash table goes more than four levels deep (I think this is unlikely). To fix this just add linear chaining at fourth level.

Please send bug reports or comments to **Robert Joop <rj@rainbow.in-berlin.de>**. See the documentation about reporting bugs and (lack of) support.

COPYING

The snacc compiler is released under the GNU General Public License. The runtime libraries are no longer under the GNU Library General Public License. The generated code is yours.

AUTHOR

Snacc was written by Mike Sample at the University of British Columbia (UBC). He used it as a tool to do encoding/decoding performance research.

It was augmented by Robert Joop at GMD Fokus with the help of some of its project partners.

ACKNOWLEDGEMENTS

This work was made possible by grants from the Canadian Institute for Telecommunications Research (CITR) and Natural Sciences and Engineering Research Council of Canada (NSERC).