An outline of the _Ruuid_ package and its required functionality. A uuid is a
16 byte (128 bit) unsigned char. There is also a 36 byte string representation.
As I read the spec neither is NULL terminated. I propose representing a uuid
in R as an S4 class. Instances have two slots, _uuid_ and _string_; both of which
will be length one character vectors. Each will contain a _CHAR_ SXP which will
be NULL terminated.

The R interface will consist of four functions that mimic the underlying C
structure.

**getuuid**  Return one object of class _uuid_.

**str2uuid**  Take a string and return the relevant uuid (as a length one character
vector).

**uuid2str**  Take a uuid and return a string.

Note that all three functions should probably be implemented using a inter-
face function. That is, R calls _Ruuid_something_, that calls the real C routine.
This will allow us to share the second set of C entry points with other libraries.
It is very unlikely that much uuid processing will happen at the R level, but it
will happen at the C level.

The windows interface is different from the Linux one. There is an exam-
than build Ts'o's library we should just link to the built in one (rpcrt4.lib).
This has (at least) one more function, UuidCreateSequential which we should
probably provide and R interface for (for Unix just make it the same as getuuid).

An interesting question is whether a uuid created under Windows (and serial-
ized via R’s mechanism) will restore on Unix and will have the same relationship
between uuid and its string representation. You might want to create two rda's
(one with a Unix version and one with a Windows version), then testing on any
platform could be designed to say whether string <-> uuid equality is preserved.