Head of repository makes core 2.0, has new services

The head of the repository for rceapp/ now makes the experimental core version 2.0.

Changes

- Tasks started with runTask from the shell now inherit the shell's definitions of the C stdio streams stdin, stdout and stderr. printf(), fprintf() et al. used from the child task now output to the telnet socket instead of to the in-memory syslog.

- If you use the version of Logger() now in RCE::service it too will output to the telnet socket. Modules run with runTask no longer need to initialize the logging package.

- New services in rce/service (namespace RCE::service):
    - EnumInfo. A template designed to make it easier to work with enumeration types. See rce/ppi/PortType.hh for an example of how to use it.
    - Logging service.
        - I've copied classes Logger, LoggerImpl and LogMessage from quarks.
        - You no longer need to initialize logging in modules run using runTask.
        - PtyLogger isn't copied, instead there is StderrLogger which is active by default.
        - Use class LoggingGuard to change the logging implementation or the severity threshold for a block of code.
    - Notepad. Each instance of Notepad tries to allocate one of the 16 notepad slots available under RTEMS; the class tracks which slots are in use. Each slot is a 32-bit value inside the task control block which therefore varies from task to task without the extra overhead of RTEMS task-variables.
    - Once. Runs an initialization function exactly once no matter how many tasks are competing for it.
    - Semaphore (Semaphore-new.hh). A re-implementation of the old Semaphore class.
    - SemaphoreGuard. Makes sure that a Semaphore is held only as long as the guard instance exists. Used to guarantee that a Semaphore is released when a block of code is exited in many places (throws, returns, gotos).
    - StringBuffer. Assembles a dynamically allocated C-string from smaller pieces, with formatting.
    - readAll(), writeAll() (rwall.hh). Used with devices like sockets which may exit from read() and write() without transferring all the data you asked them to (even when no error occurs).
    - Thread (Thread-new.hh). An abstract base class for managing tasks; you derive a class and implement the virtual function body(). New Threads inherit the C stdio streams from the creating task. If that task is also managed by Thread the new Thread can inherit the logging implementation and RTEMS priority as well. Class Thread uses a Notepad slot to keep a pointer to the Thread instance used to manage a task; the slot can be read with the static member function currentThread() even in code that is not in a Thread-derived class, e.g., library code. In your derived classes you can place extra data members which gives you thread-local storage without the overhead incurred by RTEMS' task variables. If your code fails to catch a C++ exception even at top level then the underlying RTEMS task is suspended.