R"""Node manager

Author : Anders Andersen
Created On : Mon Aug 24 00:09:36 1998
Last Modified By: Anders Andersen
Last Modified On: Fri Sep 10 13:35:11 1999
Status : Unknown, Use with caution!

This module implements the node manager. There is only one node manager on each node (host), and its main purpose is to provide communication ports to capsules and bindings.

The node manager is either started as a stand alone python program or automatically by a capsule (this is done if the capsule can not detect a node manager on the node it is running). The node manager will serve requests (normally requests for communication ports) until it receives a terminate command.

If the node manager module is loaded in another program, a node manager proxy is created and it is used to access the services of the running node manager (`nm`).

The node manager provides (through a node manager proxy) the following services: (i) `ping` is used the test if the actual node manager is responding (does exists), (ii) `newPort` allocates a new communication port on this node, (iii) `delPort` releases a previous allocated communication port on this node and (iv) `terminate` terminates the node manager.

The node manager is usually invisible for the programmer. The capsules and bindings silently access the node manager to manage their communication ports.

"""

```
# We need to check the type of some attributes
from types import *

# For low level communication
from socket import *

# Sleep
from time import *

# Mainly to grab information about exceptions
import sys

# OS related functionality (like fork)
import os

# Misc values for the Open-ORB core
from misc import *

# Message object for the Open-ORB core
from msg import *


class NodeMngrException(OpenORBException):
    R"""Node manager exception

    All exceptions or errors introduced by the nodemngr module is handled by this exception class.

    """
    pass


class NodeMngr:
```

```
R"""Node manager
```

The node manager manages all the capsules on its node (host). Its only task is to manage communication ports. The node manager is accessed through a node manager proxy.

```
"""

def __init__(self):
    R"""Initialize and run the node manager

    First the node manager initialize itself and then it goes into a loop serving requests from capsules.
    Each request is served by calling the method for the given request type (req["op"]).

    """

    # Initialize data structures
    debug("New node manager at %s created" % (gethostname(),))
    self.op = {"ping": self.ping,
               "newPort": self.newPort, "delPort": self.delPort}
    self.ports = []

    # Create and initialize socket
    self.listen = Msg("", nmPort, 1)

    # The main (serving) loop
    while 1:

        # Recieve a request
        connection, request = self.listen.recvreq()
        for req in request:
            debug("Node manager request: %s" % (`req`,))

            # Terminate?
            if req["op"] == "terminate":
                return

            # Perform the request and send a reply (possible an error)
            try:
                if not req.has_key("args"): req["args"] = ()
                if not req.has_key("kw"): req["kw"] = {}
                rep = apply(self.op[req["op"]], req["args"], req["kw"])
            except Exception:
                (exc, val, tb) = sys.exc_info()
                str = "Node manager error: %s (%s)" % (req["op"], `val`)
                debug(str)
                self.listen.sendrep(connection, ErrorObject(exc, str, tb))
            else:
                self.listen.sendrep(connection, rep)

def __del__(self):
    R"""Node manager deleted

    Clean up before the node manager is deleted.

    """
    debug("New node manager at %s deleted" % (gethostname(),))

def ping(self):
    R"""Ping

    Used to check if this node manager is alive.

    """
    return "NodeMngr at %s is alive" % (gethostname(),)
```

```
140
    def newPort(self, info):                                                    141
        R"""Register new communication port                                     142

        A new communication port is registered with the information info and a port (and index) number
        is returned.

        """
        index = -1                                                              148
        for i in range(len(self.ports)):                                        149
            if self.ports[i] == None:                                           150
                index = i                                                       151
                self.ports[i] = info                                            152
                break                                                           153
        if index == -1:                                                         154
            index = len(self.ports)                                             155
            self.ports.append(info)                                             156
        debug("Node manager new port %s: %d" % (‘info‘, firstPort + index))     157
        return firstPort + index                                                158
                                                                                159
    def delPort(self, port):                                                    160
        R"""Delete communication port                                           161

        Remove a communication port with given port number from the register.

        """
        index = port - firstPort                                                167
        if (index >= 0) and (index < len(self.ports)):                          168
            debug("Port %d (%s) deleted" % (port, ‘self.ports[index]‘))         169
            self.ports[index] = None                                            170
            return 0                                                            171
        else:                                                                   172
            str = "Node manager: Unknown port: %d" % (port,)                    173
            raise IndexError, str                                               174
                                                                                175
                                                                                176
class NodeMngrProxy:                                                           177
    R"""Node manager proxy                                                      178

    A proxy for the node manager. Call the methods of this proxy and they are forwarded to the actual node
    manager and performed there.

    """
                                                                                184
    def __init__(self):                                                        185
        R"""Initialize the node manager proxy                                   186

        Check if the node manager is up and running. If not, start it.

        """
        self.message = Msg("", nmPort)                                          191
        try:                                                                    192
            rep = self.ping()                                                   193
        except error, val:                                                      194
            debug("Node manager not running, try to start it")                  195
            try:                                                                196
                pid = os.fork()                                                 197
            except AttributeError, val:                                         198
                str = "Nodemangr: fork() not available on this platform: %s" \  199
                      % (‘val‘,)                                                200
                debug(str)                                                      201
                raise NodeMngrException, str                                    202
            if pid == 0:                                                        203
                for p in sys.path:                                              204
```

```
                    try:
                        path = "%s/nodemngr.py" % (p,)
                        file = open(path, 'r')
                        file.close()
                        break
                    except IOError:
                        pass
                os.execlp("python", "-d", path)
            else:
                sleep(2)
                try:
                    rep = self.ping()
                except error, val:
                    str = "Node manager not responding: %s" % (`val`,)
                    debug(str)
                    raise NodeMngrException, str
                except Exception, val:
                    str = "Node manager unknown error: %s" % (`val`,)
                    debug(str)
                    raise NodeMngrException, str
                if isinstance(rep, ErrorObject):
                    debug(`ErrorObject.val`)
                    raise ErrorObject.exc, ErrorObject.val
        debug("NodeMngrProxy initialized: %s" % (`rep`,))

    def ping(self):
        R"""Ping node manager

        Check if the node manager is alive. The node manager returns an alive message (text string).

        """
        return self.message.message({"op": "ping"})

    def stopserve(self):
        R"""Terminate node manager

        Send a message to the node manager asking it to terminate itself. We do not check for a result of
        this message.

        """
        self.message.announce({"op": "terminate"})

    def newPort(self, info):
        R"""Request new communication port

        A new communication port is requested. The new port number is returned and the port is registered
        under the given information (info).

        """
        return self.message.message({"op": "newPort", "args": (info,)})

    def delPort(self, port):
        R"""Delete a communication port

        Delete the communication port with the given port number.

        """
        self.message.message({"op": "delPort", "args": (port,)})


# Make an instance of the node manager or a proxy for the node manager
if __name__ == "__main__":
    nm = NodeMngr()
else:
```

```
    nm = NodeMngrProxy()
```

```
# LocalWords:  Aug Oct UK NORUT aacodefont newPort delPort misc msg nodemngr
# LocalWords:  NodeMngrException OpenORBException NodeMngr def init req op kw
# LocalWords:  gethostname nmPort recvreq args exc tb sendrep ErrorObject del
# LocalWords:  SystemExit str len firstPort IndexError NodeMngrProxy pid py
# LocalWords:  AttributeError Nodemangr execlp isinstance
```