JGS Tue Apr 13 17:59:13 PDT 1993

Backwads-Compatibility For Archived Objects

It is important for new versions of Locus to maintain reverse-compatibility (at least read compatible) with older versions, so that it can read files written by older versions. Since the files are written using NXTypedStreams, we rely on versioning information in the stream to determine how the file should be read. Each class has its own version number, which is set through the +initialize method. The version number MUST be incremented (at least) whenever the archived structure of the object gets changed (that is, when the way the the object is archived is changed by adding new instance variables, etc).

In the -read: method of each of these archived classes, it first gets the version number for the archived object. It then reads the archived object according to this version number. This is done thorugh a series of if .. then ... else loops, where there is a different if test for each different archive version that is supported. Each code block for the test

containse all the code required to read a given version. If the version number isn't recognized by any of these tests, an error occurs and the read fails.

For example, consider the class Foo. In the first version (1), it contained two integers that were archived. After other changes, it became version 5 when a string was added that also gets archived. In version 8, another string and integer were added. So, the -read: method would look something like this:

```
- read:(NXTypedStream *)stream
{
    int version;

    [super read:stream];

    version = NXTypedStreamClassVersion ( stream, [[self class] name] );

    if ( version <= 4 ) {
        NXReadTypes ( stream, "ii", &int1, &int2 );
        string1 = NULL;
        string2 = NULL;
        int3 = 0;
    } else
```

```
    if ( version <=7 ) {
        NXReadTypes ( stream, "ii*", &int1, &int2, &string1 );
        string2 = NULL;
        int3 = 0;
    } else

    if ( version <= Foo_VERSION ) { // Up thru the current version
        NXReadTypes ( stream, "ii**i", &int1, &int2, &string1, &string2, &int3 );
    } else

    { // Error: unrecognized archive version
        printf ( "Error: Unrecognized version %d of class Foo\n", version );
    }

    return ( self );
}
```

Moving Away from Depedence on NXTypedStreams

It would be nice (and probably wise) to get away from using NXTypedStreams to do the archiving. My biggest concern is if any of the standard classes on which the archived

objects are based get changed. This is especially true for the Cell class, of which ItemCell is a subclass. Folder and Group are subclasses of List, and it would be good to totally separate these classes from the standard Appkit and Common classes. Not using typed streams would allow employing a mechanism that uses flat ASCII files, which is more consistent with the paradigm for UNIX (and derivative) systems.