Representation issues in file transfer

Here are some of the problems with signing code that I described, with references. In all cases, I am talking about classes of file that can be and are transferred between systems and where code signing would be useful. The references are merely the examples I knew how to find in a hurry - zOS is used as an example of a non-Unix-like system solely because I happen to know its ancestor very well.

Text Files

Even on Unix-like systems with a high level of POSIX compliance (including Microsoft Windows), there is no standard form of end-of-line representation, and all of LF, CR-LF and CR are used. Also, separators like new-page can be LF-FF, CR-FF and almost certainly many more combinations. Once one leaves those, things get rapidly worse. Under zOS, text line separators often use ASA controls (see references 1 and 2); I know that many embedded and specialist systems are not Unix-like, but have no idea of the details.

Many or most text line formats do not regard trailing spaces as significant (see references 3, 6 and 8), and many editors, file transfer tools and mail systems remove or add them incidentally. The same often applies to the remapping of tabs to multiple spaces and vice versa, and sometimes even to line wrapping (see reference 8). Reference 8 also refers to the 'signature separator' ('-- ') and the problems it causes.

In many character sets, including the now-dominant Unicode (see reference 9) many characters may have multiple, equivalent representations, and file transfer tools are at liberty to convert between them. It gets much worse once you need to translate between very different character sets (e.g. EBCDIC and Unicode), where the interpretation of a text file and therefore the corresponding Unicode characters can depend on options in the application using it (see reference 10).

Despite this, those files can be and are transferred transparently between systems using appropriate translation (references omitted as widely known).

Binary files

Many binary file formats on both hardware are not simple streams or records of bytes, and the details can change when they are copied between devices (on the same system) or between systems. Examples of this that I am familiar with include Fortran unformatted files (and direct-access ones), almost all file formats under zOS, and many magnetic tape formats (see references 1 and 3).

Some hardware devices (such as magnetic tapes) and some file formats (see reference 7) use fixed-size blocks with optional padding (usually bytes of zero, but sometimes undefined). Many binary file formats use record boundaries (e.g. Fortran unformatted files - see reference 3), and their representation varies between system (or even within the same system - see references 1 and 5).

Where the differences are relatively simple, such as the endianness of integers or floating-point numbers, insignificant padding, or different record boundary indicators, those files can be and are transferred transparently between systems using appropriate translation (see reference 5).

References:

1: http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp

Select "z/OS concepts" and then "z/OS storage constructs: File systems, data sets, and more"

2: http://publib.boulder.ibm.com/infocenter/zos/v1r13/index.jsp

Select "z/OS XL C/C++ Programming Guide", "Input and Output" and then "Using ASA text files"

3: ISO/IEC 1539-1:2010 (Fortran)

4: http://www.ietf.org/rfc/rfc959.txt

5: http://gcc.gnu.org/onlinedocs/gfortran/Runtime-Options.html

6: ISO/IEC 9899:2011 (C) and ISO/IEC 14882:2011 (C++)

7: http://www.gnu.org/software/tar/manual/html_node/Standard.html

8: http://www.ietf.org/rfc/rfc3676.txt and http://www.ietf.org/rfc/rfc3676.txt

9: http://www.unicode.org/versions/Unicode6.2.0/

See 2.12 "2.12 Equivalent Sequences and Normalization".

10: I couldn't find a simple one. A search for

site:http://publib.boulder.ibm.com/infocenter/zos/* "code pages"