```
The file name is "Test"
```

In the Printf command line, \" embeds a double-quote character in the format string. If you omitted the backslash, the " would end the format string. The \r specifies that you want a carriage return in the format string.

Unicode Escape Sequences in Strings

Igor represents text internally in UTF-8 format. UTF-8 is a byte-oriented encoding format for Unicode. In this section we see how to use the \u, \U and \x escape sequences to represent Unicode characters in literal strings.

The use of UTF-8 and these escape sequences requires Igor Pro 7.00 or later. If you use them, your will get incorrect data or errors in Igor Pro 6.x.

In the first set of examples, we use the hexadecimal code 0041. This is hexadecimal for the Unicode code point representing CAPITAL LETTER A.

```
String str
str = "\x41"          // Specify code point using UTF-8 escape sequence
Print str
str = "\u0041"        // Specify code point using UTF-16 escape sequence
Print str
str = "\U0000041"     // Specify code point using UTF-32 escape sequence
Print str
```

When you use \xdd, Igor replaces the escape sequence with the byte specified by dd. Therefore you must specify byte values representing a valid UTF-8 character. 41 is hexadecimal for the UTF-8 encoding for CAPITAL LETTER A. If the code point that you specify is not a valid UTF-16 or UTF-32 code point, Igor replaces the escape sequence with the Unicode replacement character.

When you use \udddd or \Udddddddd, Igor replaces the escape sequence with the UTF-8 bytes for the corresponding Unicode code point as represented in UTF-16 or UTF-32 format.

Now we specify the characters for "Toyota" in Japanese Kanji:

```
str = "\xE8\xB1\x8A\xE7\x94\xB0"       // UTF-8
Print str
str = "\u8C4A\u7530"                   // UTF-16
Print str
str = "\U00008C4A\U00007530"           // UTF-32
Print str
```

In order to see the correct characters you must be using a font that includes Japanese characters.

\U is mainly of use to enter rare Unicode characters that are not in the Basic Multilingual Plane. Such characters can not be specified with single 16-bit UTF-16. Here is an example:

```
str = "\U00010300"                     // UTF-32 for OLD ITALIC LETTER A
Print str
str = "\xF0\x90\x8C\x80"               // UTF-8 for OLD ITALIC LETTER A
Print str
```

OLD ITALIC LETTER A is located in the Unicode Supplementary Multilingual Plane (plane 1).