Dump (HexEdit)

Kaminski / cs3310

Dump a plain ASCII (text) file to see:

- which physical record (or "a Line"-indicator) storage method is used on THIS computer system (which is invisible to the text-editor and text-file printer utility): the line-separator method or the preceding byte-count method?
  - and for the line-separator method, what separator is stored when the user hits the "ENTER" key in a text editor: a <LF> or both <CR><LF>?
- what was actually written to the file when the program supposedly "wrote out a record" to the file?
  - and is it in the correct location? And are the record contents correct?
- what value and number of bytes are actually stored for "unprintable" ASCII char's (like a TAB)?

---------------------------------- W I N D O W S -----------------------------------

Windows uses a line-separator approach --- with `<CR><LF>` (i.e., `\r \n` or `0D 0A`) for the separator.

[The file as created & viewed in Notepad

```
0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SPACE TABabc
```

The HexEdit program display

```
            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F     0123456789ABCDEF
00000000   30 31 32 33 34 35 36 37 38 39 0D 0A 41 42 43 44     0123456789..ABCD
00000010   45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54     EFGHIJKLMNOPQRST
00000020   55 56 57 58 59 5A 0D 0A 20 53 50 41 43 45 09 54     UVWXYZ.. SPACE.T
00000030   41 42 61 62 63 0D 0A                                ABabc..
```

[- OFFSET -] [- HEX interpretation. of bytes -] [- - - - - - CHAR interp. - - - -]
(byte counter) *(. indicates "unprintable char)*

----------------------------- S U N (U N I X) ---------------------------------

Sun/Unix uses a line-separator approach --- with a `<LF>` (i.e., `\n` or `0a`) for the separator.

```
SUN> cat myfile.txt
0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SPACE TABabc
```

[`cat` is a system utility to type out a text file]

[`od` for "octal dump", `-c` for the character interpretation instead of octal]

```
SUN> od -c myfile.txt
0000000    0   1   2   3   4   5   6   7   8   9  \n   A   B   C   D   E
0000020    F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U
0000040    V   W   X   Y   Z  \n       S   P   A   C   E  \t   T   A   B
0000060    a   b   c  \n
```

*[OFFSET in OCTAL]^^^^^^^^^^^^*
(byte counter)

```
SUN> od -x myfile.txt
0000000 3031 3233 3435 3637 3839 0a41 4243 4445
0000020 4647 4849 4a4b 4c4d 4e4f 5051 5253 5455
0000040 5657 5859 5a0a 2053 5041 4345 0954 4142
0000060 6162 630a
```

[`-x` or hexadecimal dump]

SUN> od **-Ad** filename *[`-Ad` for decimal byte-counter on left vs. octal, the default]*

SUN> od **-xc** filename *[`-xc` for both char and hex dumps integrated together]*

SUN> od **-cw18** filename *[`-cw18` for show 18 bytes per line vs. default 16 (e.g., for alignment of fixed-length records)]*

----------------------------------- LINUX -----------------------------------

Linux uses a line-separator approach --- with a `<LF>` (i.e., `\n` or `0a`) for the separator

```
Linux> cat myfile.txt                                        [cat  – a system utility to type out a text file]
0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SPACE TABabc
```

[NOTE: Linux is similar to Unix in its commands and option specifiers].

```
Linux> od -c myfile.txt
0000000   0   1   2   3   4   5   6   7   8   9  \n   A   B   C   D   E
0000020   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U
0000040   V   W   X   Y   Z  \n       S   P   A   C   E  \t   T   A   B
0000060   a   b   c  \n
[OFFSET in OCTAL]^^^^^^^^^^^^
 (byte counter)
```

```
Linux> od -x myfile.txt
0000000 3130 3332 3534 3736 3938 410a 4342 4544
0000020 4746 4948 4b4a 4d4c 4f4e 5150 5352 5554
0000040 5756 5958 0a5a 5320 4150 4543 5409 4241
0000060 6261 0a63
```

[NOTE: The dump's CHARACTER interpretation prints out the text file characters in the same order as the typed out (`cat`) version.
However, dump's HEX interpretation prints out the file's bytes reversing the byte-pairs because of the "ENDIAN issue" – see below
e.g., the char's `'0'` and `'1'` have ASCII codes 30 and 31 – so `01` in the CHAR file dumps as `3130` in the HEX dump].

----------------------------------- VMS (on VAX) ------------------------------

VMS uses a line byte-count approach (preceding each line with the count), but no `<CR><LF>` separators to be taken literally by `type`.

```
VMS> type myfile.txt                                         [type    – a system utility to type out a text file]
0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SPACE TABabc
```

```
VMS> dump myfile.txt
4241001A 39383736 35343332 3130000A   ..0123456789..AB   000000
5251504F 4E4D4C4B 4A494847 46454443   CDEFGHIJKLMNOPQR   000010
45434150 53200006 5A595857 56555453   STUVWXYZ.. SPACE   000020
00000000 0000FFFF FF636261 42415409   .TABabc.........   000030
...
00000000 00000000 00000000 00000000   ................   0001F0
[----------- HEX interp. of bytes, right-to-left ---- -------]   [--- CHAR interp.L-to-R -----]   [OFFSET (byte count) in HEX]
```

```
VMS> dir/full myfile.txt
...                                   [NOTE: with VMS, file attributes are stored as PHYSICAL characteristics in OS]
File organization:  Sequential
Record format:      Variable length, maximum 26 bytes
Record attributes:  Carriage return carriage control
```