JASWORD

(another word processor for the C64...)

Features:

- real print view with 80 character screen
- smooth working pace, even with long texts
- Paragraph layout with 10 freely definable formats
- Print design integrated in the text
- Multi-column capability

Hardware requirement:

A good monitor and a video cable with 8-pin DIN connector for Luma+Chroma+Audio (also called Y/C cable or S-Video). Video cables with 5-pin DIN plugs on the C64 and two cinches on the monitor (composite or FBAS) give a lousy picture!

Program operation:

The text is edited with key commands, there is an overview in menu item f1. Further functions are available in the menus f2...f8, the status bar provides information about input options. Just look, further explanation is unnecessary in most cases.

Text representation:

The text is divided into main text, headers and footers. Text in headers and footers is identical on all pages, with the exception of the page number (see wildcard). In the first, unfinished page, the footers can only be reached with the cursor down function C= CRSR.

The representation assumes a fixed page width of 80 characters. Margin distances can be specified in menu f3, the color of the margins can be specified in f2.

Text layout:

The text formatting always refers to entire paragraphs. 10 formats can be created in menu f3, changes immediately affect the entire text. A text format is defined by five parameters:

- Name (appears in the status bar at the bottom right)
- Alignment (left, right, center, justified)
- Distance from the left edge of the page in the first line
- Distance from the left edge of the page in the following lines
- Distance from the right edge of the page

The assignment in the text is done with the keys CTRL 0...9.

Text marking:

Marking is started end to end with the STOP key. Shorter or longer sections of text can be marked for the following operations (see f1):

- Extinguish
- Copy at the cursor position as often as you like
- move to cursor position
- Assign paragraph format
- Assign font style (character accurate, for long sections may be time-consuming)

Menu operations with marked sections:

- f6 save text section (see file management)

- f4 sum (see extras)
- f4 sort (see extras)

If the entire text is marked with SHIFT STOP, this also includes headers and footers. Some of the functions mentioned are then not available.

Attention: If a marking is set, the format and font style always refer to the marked section!

Multi-column text:
The program is prepared to display and print up to 10 columns of text side by side. In order to create multi-column text, the formats must first be prepared in the f3 menu. The columns must be set up in such a way that they have space next to each other with at least 2 characters between them. (Note that the specification of the right margin refers to the distance from the right edge of the page.)

When entering multi-column texts, the following rule applies: if the following paragraph has room at the right of the previous paragraph due to its margin setting, then it will be displayed there.

Example of two-column text:
Column 1 starts on the left. The right edge of the column leaves ample room for one or more right columns.

Column 2 starts at least 2 characters to the right of the previous column. The line break occurs within the columns and the columns can be edited independently.

To end a column section or start a new one, a format is selected for the next paragraph that does not fit on the right side of the column.

Print control characters in the text:
For emphasis, 6 different font style printing instructions (**double width**, **bold**, *italic*, underline, superscript and subscript) can be anchored in the text itself. The control codes of these instructions are to be set in menu f8 in the print code table. All font styles always appear on the screen simply as underlining, the type of emphasis can be seen in the status bar below when the cursor moves to the relevant text passage.

In addition, free print control characters can be defined, which, however, appear in the text as control characters (letter with overline):

- (CTRL A) freely assignable, e.g. orange ribbon color
- (CTRL C) freely assignable, e.g. cyan ribbon color
- (CTRL E) freely assignable, e.g. green ribbon color
- (CTRL N) intended for the NLQ statement
- (CTRL X) freely assignable, e.g. black ribbon color

If a control character is to apply to the entire text, it can be written in the initialization sequence menu f7. Integrated font style printing instructions cannot be combined (e.g. italics and bold); free print control characters must be used for such combinations.

The wildcard (CTRL =):
The wildcard performs various tasks:

- In headers or footers the current page number (1-3 digits without leading zeros) is printed.
- In the main text, when printing, a file can be opened on disk in order to read in external data. This can be used to load data for a mail merge or graphic data for the printer (see Mail Merge/Graphic Printing).
- The wildcard can be used in the Menu f4 Find/Replace input field as an excluded character.

File management:
In addition to the text, the Jasword text files contain all the current menu settings (including conversion and print code tables). Text files are saved as type PRG, since they can actually be executed in BASIC: You can load them like a program and start them with RUN, JASWORD will then be loaded automatically.

Otherwise, by default, the program searches for a file named "template 0" on the disc when it is started and loads it if it exists. Any text file with preferred menu settings or text templates can be saved under this file name.

In addition to a complete text file, sections of text can be marked, saved on disk and reloaded from disk. No menu settings are saved, but the format of the text sections is retained, provided no conversion table is used. In principle, it doesn't matter whether text sequences are saved under the file type SEQ or PRG.

When loading from directory by selecting a file, the program automatically determines the file type and file format of text sequences. If it is a foreign format, the current conversion table is used when loading.

If the entire text is selected with the SHIFT STOP combination and saved as a text sequence, this also includes the text in headers and footers. Such a text sequence cannot be loaded anywhere in another text, but only in a completely empty text file.

General information:
- File names may not be longer than 16 characters when entered. The input fields are 18 characters long, but only to be able to integrate trailing spaces into the file name with the help of quotation marks.

- In the disc directory there is the possibility of previewing any file. Unformatted text is shown, external files are translated using the conversion table.

- If the program was terminated by mistake, the text file still in memory can be saved by loading the file "restore" and starting it with RUN.

Conversion table:
The conversion table is intended for import/export of texts from/to other applications. A code can be defined for each character in menu f8. Please note that each code must be confirmed with RETURN, otherwise the change will not be saved. When saving text sequences using the conversion table, the pure text is converted, the layout is lost. When loading text sequences in a foreign format, an analogous translation into the program-internal code takes place.

The conversion table can be saved as a separate file in menu f8 and loaded as required. This makes it easy to switch conversion tables for common import/export operations. Complete conversion tables for PET-ASCII, BS-CODE and ISO8859-1 (PC) are available. Most applications will use one of these three code types for the text. It may then be important to find out which code is used as the paragraph mark. In the conversion table, this code is the last definable character, denoted as CTRL R (with an overscore) for RETURN. (For example, Vizawrite uses BS code for the body and code 220 for the paragraph mark.)

Loading texts using the conversion table can take quite a long time, since the imported text is processed character by character, just like input from the keyboard. Before loading unknown files in this way, you should use the preview function in the disc directory to check whether they are text files and whether the appropriate conversion table is loaded.

Printer setting:
The "Print from/to page" fields are created automatically when menu f7 is called up and can be modified before printing if necessary. When printing longer documents that have been divided into individual text files for storage reasons, the start of page numbering for printing (see wildcard) can be specified. An additional left margin can be useful if the printer allows more than 80 print columns. So 80 columns can be used for pure text, the margin is added when printing.

Two fields are provided for printer initialization, which are sent to the output device with their respective secondary addresses. For example, before the actual printout, you can send a printer setting command via a specific secondary address. The content of the fields can consist of text or control characters or remain empty. The control characters CTRL A,C,E,N,X, the control characters for the font styles CTRL D,B,I,U,H,L and the control character CTRL R, which represents the line break "Carriage Return" CR, can be used. The actual text printout is sent via the channel that was opened with "Initialize output".

Since the device number can be freely selected between 4 and 15, a printout can be sent to a diskfile as well as to a printer. For example, the old file can be deleted with "s:printout" secondary address 15 and the printout in a file can be effected with "printout,s,w" secondary address 2. Unlike exporting a text sequence using the conversion table, the printout is fully formatted (including multi-column text) and wrapped with a CR at each line.

Print code table:
This table serves the same purpose for the printout as the conversion table for the file export: The internal character code is translated into a freely definable print code. Control characters can be assigned a code sequence of 4 bytes, suitable for the usual ESC sequences that start with code 27 and then contain 1-3 bytes of instructions. (Leading 0 bytes in the sequence are skipped on output.)

The print code table is defined or loaded in menu f8. Complete print code tables for CBM-DIN and ISO8859-1 (PC) are available, the control characters may have to be adapted to the respective printer.

Serial/graphic printing:
This option allows data to be loaded from the disc during printing and inserted at designated points in the text. The wildcard character (CTRL =) in the main text is used for this. In menu f7 you can specify

whether this option is to be used and, if so, whether text or code is to be loaded. The main difference is that code is passed 1:1 to the printer, while text is converted using the print code table.

Jasword text files, text sequences or third-party files can be used as data sources. Text data can be divided into individual data elements of variable length by a separator. Setting a separator is only taken into account when loading text, the separator has no effect with code. This now gives rise to various possibilities:

- If you want to load text in fields of fixed length, you put a series of wildcards in the desired Field length in a row and set 1 byte per wildcard. If the source data is not also of fixed length, a seperator between the elements can be defined. If the seperator is reached when reading the source data, the remaining columns in the field are filled with spaces.

- Should the data import flow seamlessly into the text, one uses only 1 wildcard character per data element and sets an arbitrarily high, but in any case sufficient number of bytes per wildcard. The reading off the data source in this element is stoped, when the seperator has been reached.

- The loading of code enables e.g. the transmission of graphics to the printer. A field of wildcards is placed in the text approximately in the size of the graphic, and an appropriate value of bytes per wildcard is choosen. Of course, this requires that the graphics data are present in a directly usable form for the printer.

Once the source data file has been read, further wildcard characters in the text are ignored and no further copies of the text will be printed.

Extras:
The search function in menu f4 is limited to the main text, headers and footers remain excluded. Search and replace can be interrupted at any time, the cursor then remains at the last place found. With the replace option, you can decide at each finding whether to replace or continue searching. The search is case-sensitive. The wildcard (CTRL =) represents an excluded character. This characters can also be excluded from replacement if the wildcard is used in the replace field. The exact position does not have to match the search field, but the number of wildcards does.

The other options in the f4 menu, summing and sorting, only refer to previously marked sections of text. Both functions expect single-line paragraphs - i.e. lines that end with a paragraph mark (CR / RETURN). Multi-line paragraphs do not cause an error, but only the first line in the paragraph is evaluated.

When summing, the marked lines are searched for the first numeric character (number, sign or point), the value found after that is added to a sum and entered at the end of the operation at the current cursor position. It does not matter if there is text in front of the numerical value, but it must not contain a minus (hyphen) or period. And only one numeric value can be processed per line. The summation in multi-column texts works well; of course each column must be separately marked and totaled.

When sorting, paragraphs are ordered according to their alphanumeric

value.

BASIC:
A somewhat strange feature is the direct access to the C64-BASIC: Each paragraph, up to a maximum length of 80 characters, is passed to the BASIC interpreter when you press CTRL RETURN, and its output is included in the text. One obvious use is to use BASIC for calculations - who likes sitting at a computer when you need to pull a calculator out of the drawer to do some math? You can also run small BASIC programs and even save and load them. Small programs because only 1021 bytes are available for BASIC. The thing is not intended for anything more than small calculation programs or routines that create, for example, a strictly schematic form. Other limitations and special features of basic access are:

- Commands are entered in lower case letters, upper case letters are of course allowed for texts in quotation marks.

- It is not possible to enter control characters in quotation marks, but some control characters can be used as CHRS() codes:

  chr$(29) CRSR to the right
  chr$(157) CRSR to the left
  chr$(17) CRSR down
  chr$(145) CRSR up
  chr$(20) DEL
  chr$(148) INST
  chr$(19) HOME (top of page)
  chr$(147) CLR (delete a line)
  chr$(13) RETURN (end of paragraph)
  chr$(141) SHIFT RETURN (jump to next line)

- The BASIC output within an existing text is not without complications, since the interpreter e.g. outputs a CRSR right after numbers, which may cause a jump to the next paragraph.

- The BASIC command INPUT from the keyboard doesn't work (it returns an empty variable).

- Access to datasette or RS232 is prohibited, as the associated system registers are used by the program. However, access to disc is possible without any problems. (One can e.g. implement a program-controlled input from a text file.)

- It is better not to try POKEs that work in the normal BASIC environment (e.g. setting the cursor), as many registers are assigned differently and the program may crash.

- A running BASIC program is not interrupted with the STOP key but with the left arrow key. A CONT after abortion is not possible.

BASIC system messages are displayed in the status bar, but it is not necessary to confirm the message.

Memory requirements:
The main memory is exactly 32 kilobytes in size. How much text has space in it cannot be specified exactly in pages and lines, since the memory requirement depends on the formatting. In the case of loosely written texts with margins, maybe 10 pages have space, in the case of densely written texts and full width, correspondingly less. In order

for insertions in long texts to proceed at an acceptable speed, the program needs some free space in memory. As free memory approaches zero, input processing may become increasingly sluggish.

Program History:
By using Novaterm 10r2 I got to know the 80-character display on the C64 for the first time. After I soldered the right video cable, I was pleasantly surprised by the image sharpness and readability. So the desire for a suitable word processing program grew. I've tried various programs that can be downloaded free of charge, without being really happy. There are other supposedly good programs that I haven't tested because I couldn't find them.

The name "Jasword" is an abbreviation for "Just Another Stupid WORD processor", and a reference to "TASWORD", which came closest to my ideas of the 80-character programs tested. The program was created in 2004 and the naming expresses the fact that a new word processing program for a 20-year-old computer probably does not represent a significant contribution to progress. On the other hand, I had a lot of fun developing it, and the project is ecologically and socially harmless, which is enough of a plus point for me ;-)

Copyright:

Included in the program is Pasi Ojala's Pucrunch Decompression routine. For the English version Program text and info file were translated by d3bug.

Contact:
The most recent version will always be available at the following address: http://www.webnet.at/c64/jasword.htm
I welcome any feedback or inquiries regarding the program.
e-mail: kottira@webnet.at