

pico EDITOR

Contents:

Fundamentals	page	1
Cutting and Copying	page	2
Exiting and Saving	page	3
Searching and Spellchecking	page	4
Justifying and Unjustifying	page	5
Keystroke Summary	page	6


Pico is an acronym for pine composer, the user-friendly editor used with pine mail. A plain editor with a limited number of features, *pico* is a simple way to enter text in the HP Unix environment, whether you are creating programs, mail messages, or simple text files.

Getting Started

Enter *pico* at your Unix prompt; the *pico* editing space will appear. Just type in your document. Two *Help* lines at the bottom list *Control*-key combinations for common tasks.

Using the *Control* Key with Other Keys

In this handout: ^ stands for the Control Key:
^K means *Control-K*: Hold down  and tap 

On the *Help* lines, on the *Help* screens, and usually in this handout, the use of the *CONTROL* key is designated by the caret symbol (^), so that *Control-X* is written as ^X. Marked *Ctrl* on the keyboard,  is used in combination with other keys. Some dexterity is required to use it properly.

Control-X (written ^X) means:

- HOLD DOWN  and
- While continuing to hold down  tap (don't hold down) 

Avoiding Problems



These keys do not paginate in *pico*. They might scroll your screen back to what was displayed before you got into *pico*! Instead, use ^Y and ^V (See *Moving Around* at the bottom of this page.)



^S

This combination appears to freeze the screen.

^Q

This combination unfreezes the screen.

Getting Help





^G

gets into *pico Help*


^X

exits *pico Help* and returns to the *pico* editing screen

Moving Around

The , , , and  functions on the numeric keypad are not supported in *pico*. However, the following keystrokes move the cursor within the *pico* editing space:

Arrow keys move the cursor up, down, left, right. (Check that  is off.)

^A




moves the cursor to the beginning of the line

^E

moves the cursor to the end of the line

^2

advances the cursor one word at a time (use *only* the 2 key in the number row)

Note: SSH Secure Shell requires holding  and  keys while you strike 



^V

moves the cursor to the beginning of the next screen (advances to the next page)

^Y

moves the cursor to the beginning of the previous screen (goes back one page)

Deleting Text


-  (*Backspace* key) or
-  Either key deletes the letter to the left of the cursor
- ^D** deletes the character the cursor is on
- ^K** deletes the line the cursor is on; (stands for *Kut*)




Cutting, Copying, or Moving a Line






- ^K** cuts (deletes) the line that the cursor is on
- ^U** Uncuts (pastes back) what you have just cut
Make multiple copies by using **^U** several times: it always restores the last text that you used **^K** to cut.

Cutting and Pasting Blocks with ^6 and ^K

In order to delete or to move a section of text quickly, you must mark it first.

- Place the cursor at the beginning of the area you want to mark
- Press **^6** (Use the  in the number *row* of the keyboard, *not* the numeric keypad on the right of the letter keys).

Note: SSH Secure Shell Client requires the use of three keys:  +  + 

- When you have used **^6** to turn on marking mode, the message [Mark Set] appears above the *Help* lines at the bottom of the screen
- In this mode, when you move the cursor a highlight extends to show the text you have marked. Use any of the cursor movement keys to extend the highlight quickly; see *Moving Around* on page 1.
- When you have highlighted what you want, cut it with **^K**
- To cancel marking mode without cutting text, use  +  to toggle the command off. (Look for the [Mark UNset] confirmation.)  +  +  with SSH.

To copy the section of text that you have cut:

- Immediately after you have used **^K** to cut the text, restore it to its original position with **^U**
- Move the cursor to any other locations where you want another copy of the text and Uncut it with **^U**. The text is inserted after the position of the cursor.
- You can continue to make multiple copies until you use **^K** to cut something else (*pico* only has one level of undeleting: it stores cuts in a single buffer).

To move the section of text that you have cut:

- Move the cursor to where you want to place the text you just cut
- Press **^U** to paste (Uncut) it after the cursor position.

Saving Your Work

- ^O** saves your text without getting out of the *pico* editor; (stands for *Write Out*)

Naming Your Files

The first time you save your work, the prompt to name the file appears: **File Name to write:**

Enter a name that will describe the content of the file to you. Make names long enough to mean something to you but short enough to be easy to type.

You can use letters, numbers, and a few special characters but no spaces in your filenames. Aside from letters/numbers, restrict yourself to the period (.), the dash (-) or the underscore (_).

IMPORTANT: Unix is case sensitive; so upper- and lowercase letters in filenames are different. The names *Program1* and *program1* would refer to different files. Examples:


Good file names:	program1_homework	Doc45.cis161
Bad file names:	file*2A	(contains the illegal character *)
	prog\$2\B	(contains the illegal characters \$ and \)
	X	(doesn't tell you anything about what's in the file)

Exiting *pico*/Saving Changes: ^X

^X gets you out of the editor (stands for *eXit*)

If you have made any changes since your last edit, you get the prompt:

Save modified buffer (ANSWERING "NO" WILL DESTROY CHANGES) (y/n)?

- Tap **Y**, meaning "Yes, I want to save my changes"; or tap **N**, meaning "No, I don't want to save the changes I made since I last saved."
- The system prompts with the last name (if any) by which you saved your material:
File name to write: oldfile.name
- If you haven't given a name to the file, enter a name at this prompt; or if you want to save this version as a different file, enter a new filename.
- When the name displayed is what you want, just tap ; if not, backspace over the old name and type in a new one.

Editing an Existing File with *pico*

You will often start an assignment, save it, and come back at another time to continue editing your work. To start the *pico* editor with the file you want to edit, follow these steps:

- Use the Unix commands **ls** or **ll** to list the names of files in your directory.
- Exact spelling is important: when you tell *pico* to edit a file, you must type in the filename exactly: match the spelling down to upper- or lowercase letters.
- If you misspell a filename, *pico* will display nothing. (It assumes you are creating a new file). Don't panic (yet); try *Method 2* below.

◆Method 1: Start *pico* with a filename

- Enter the *pico* command followed by a space and the name of the file you want to work on. For example, to continue working on a file named *Program1* you would enter:

```
pico Program1
```
- If you have spelled the name of the file correctly, the editor will appear with the cursor at the beginning of the work that you saved as *Program1*.
- You can now continue to edit (don't forget to save often!).

◆Method 2: Start *pico* empty, then read in an existing file


- Use this method to call up *pico* first and then to read in a file that you have saved.
- Call up the editor from a UNIX prompt by entering:

```
pico
```
- Inside the editor, use the command **^R** (to **Read** in a file)
- At the prompt *pico* gives, enter the name of the file that you want.
- If you have spelled the filename correctly, the contents of the file will appear.
- If you haven't spelled the name of your file correctly, the following message appears highlighted above the *Help* lines at the bottom of the screen:

```
[ No such file: <your file name> ]
```


Just try **^R** again at this point — and spell the filename correctly.

One complication with Method 2:

- If you read in a file with **^R**, you will have to re-enter the name of the file the first time you save it in this *pico* session; or if you prefer to keep an unaltered copy of the original file, save your edited file with a different filename.
- When you try to save the file, (if you have spelled the filename correctly) you will get a message indicating that the file already exists, and asking if you want to overwrite it; if you want to replace the old version with the new version, reply by tapping **Y** for yes.
- After the first save, the filename appears on the top line, and when you save, that filename appears as the default name to save under: just tap  to save later modifications under the same filename.

Searching for Text with ^W


To find a word or set of characters in your text:

- Press **^W** (the *Where is* command).
- Type in what you want to search for at the `Search:` prompt, and press .
- The cursor will move to the next occurrence of what you wanted to search for.


Searching for the Same Text a Second Time

After the first time you use **^W** during a session you will see the last word(s) or letter(s) you tried to find displayed inside the **^W** prompt. For example, if you last searched for the word “astronomy,” the next time you press **^W** you will see:

```
Search [astronomy] :
```



If you just press  at this point without entering a new word to search for, you will automatically search for the next occurrence of the word “astronomy” in the document.

Checking Basic Spelling with ^T

Crude compared to the spellers in today's word processors, **^T** simply highlights and offers for editing anything looking like a word that is not in its dictionary. It points to many correctly spelled words of which it is unaware as well as most technical terms, proper names, and abbreviations. If you know that the highlighted word is correct, just press  to bypass it.

This works fine for simple typing errors. But the speller offers no suggestions for correcting the spelling: *you* have to know what corrections to make. The best advice is to have a dictionary handy.

Here is the process for running the speller:

- With the cursor anywhere in your *pico* editing space, press **^T**.
- The speller will point to the word it wants to question and give the prompt
Misspelled word:
- To accept what is displayed without making a change, just press .
- You can also backspace over the characters displayed and change the spelling:
- Then strike  to substitute the text in the file with what you entered.
- Cancel a spellcheck at any point with **^C** (stands for *Cancel*)

Locating Current Position with ^C

Some compilers indicate the line number location of errors or problems in your program code. When your program is in the *pico* editing space, you can use **^C** (the *Current Position* command) to find out the number of the line where your cursor is located.

The response displays above the *Help* lines and indicates other aspects of the cursor position.

```
[ line 8 of 16 (50%), character 218 of 2541 (8%) ]
```

In the example above, the response indicates that the cursor is positioned:

on the 8th line of a total of 16 lines in the document

(about 50% of the way through all of the lines of the document)

on the 218th character of a total of 2541 characters in the document

(about 8% of the way through all of the characters of the document)

Justifying and Unjustifying with ^J and ^U

When you type a paragraph in *pico*, the program does its version of wordwrap: when you type sentences longer than the width of the screen, the words "wrap" to the next line:

```
However, pico is not a real word processor. If you go back to edit a
paragraph you will find that adding or deleting words can break up your
paragraph into lines of uneven length.
```

For example, here is what happens when the above paragraph has the words "If you go back" changed to "and if you should want to go back":

```
However, pico is not a real word processor, and if you should want to go
back to edit a
paragraph you will find that adding or deleting words can break up your
paragraph into lines of uneven length
```

The first line gets split and creates two uneven lines. To restore the look of the paragraph, place the cursor inside it and use the command ^J (to *Justify*); this is the result:

```
However, pico is not a real word processor, and if you should want to go
back to edit a paragraph you will find that adding or deleting words
can break up your paragraph into lines of uneven length
```

If you hit ^J by mistake, or just don't like the way the justified paragraph looks, use ^U (for *Unjustify*) to undo the justify operation immediately after you have pressed ^J while the message

```
[ Can now UnJustify! ]
```

is displayed. If you move the cursor, the message disappears, and ^U goes back to its normal function as the *Uncut* or *Paste* key.

Note: A paragraph (the amount of text that ^J works on) is defined as text delimited by blank lines. If you don't have blank lines before and after what you want to justify, the ^J command may work on more text than you want.

Justifying with ^J (Not for programmers)

Note: Justification should not be used when writing programs, where you want individual command lines to remain separate. To *pico* "justifying" means putting lines together to form nice-looking paragraphs without extra white space. This is useful if you are composing a mail message or other text, but not for programming code.

Keystroke Summary for the HP Unix *pico* Editor

- ^A moves the cursor to the beginning of the line
- ^D deletes the character the cursor is on
- ^E moves the cursor to the end of the line
- ^G gets into help
- ^J justifies a paragraph of text delimited by blank lines
(^U unjustifies if used immediately afterwards)
- ^K cuts a line (or text that was highlighted using ^G)
- ^O saves (without exiting *pico*)
- ^R reads in a file (that has been saved previously)
- ^T checks spelling
- ^U pastes cut text at the position of the cursor
- ^V advances the cursor to the next screen
- ^W searches for text
- ^X gets out of *Help* if user is in *Help* mode
exits *pico* if user is not in *Help*
- ^Y moves the cursor back to the previous screen

Special Keystrokes

- ^2 moves the cursor forward one word

(You *must* use the **2** in the number *row* near the top of the keyboard, *not* on the numeric keypad)

Note: In SSH Secure Shell Client, hold both **CTRL** and **SHIFT** and then strike **2**

- ^6 prepares to mark text for blocking or highlighting

(You *must* use the **6** in the number *row* near the top of the keyboard, *not* on the numeric keypad)

Note: In SSH Secure Shell Client, hold both **CTRL** and **SHIFT** and then strike **6**

- ^B moves the cursor *Back* one character
- ^C In addition to the documented function ("Show cursor position"), this keystroke combination can *Cancel* a current operation. (For example: this keystroke combination can get you out of *Help* or interrupt the *Spell* or *Search* operations)
- ^F moves the cursor *Forward* one character
- ^H deletes the character to the left of the cursor; like **←** *Backspace*
- ^M adds a newline: breaks a line; like **↵**
- ^N moves the cursor to the *Next line*; like **↓** *Down Arrow*
- ^P moves the cursor to the *Previous line*; like **↑** *Up Arrow*
- ^S freezes the display of keystrokes; (Unfreeze with ^Q.)

Dangerous keys:

^S **Your screen is frozen!**

Hitting this combination makes the screen appear to freeze. If you use ^S, strike other keys, and then unfreeze the screen with ^Q, those "other" keystrokes will be executed after the screen comes back to life. So be careful with this!

PG UP and **PG DN** :

These don't work in *pico*. They may scroll your screen back to what was displayed before you got into *pico*! And you might lose track of your cursor. Probably not what you want to do. Use ^Y (for Previous Page); and ^V (for Next Page)