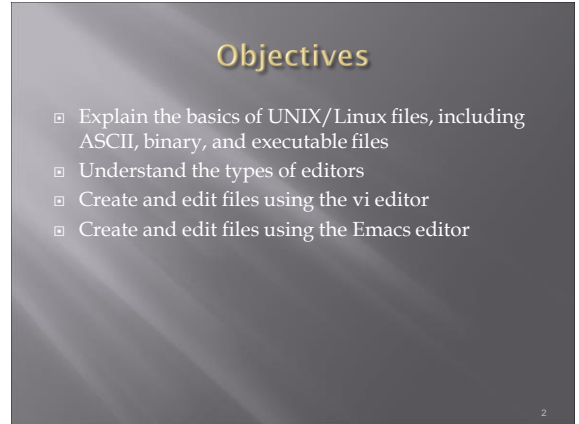
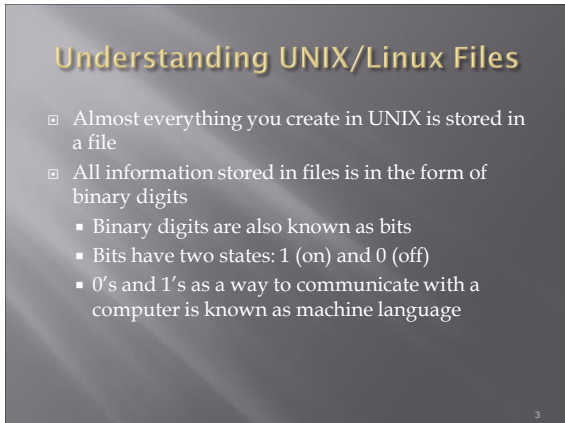


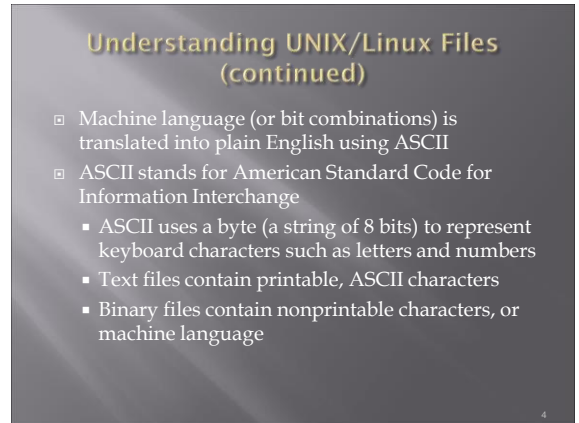
1



2



3



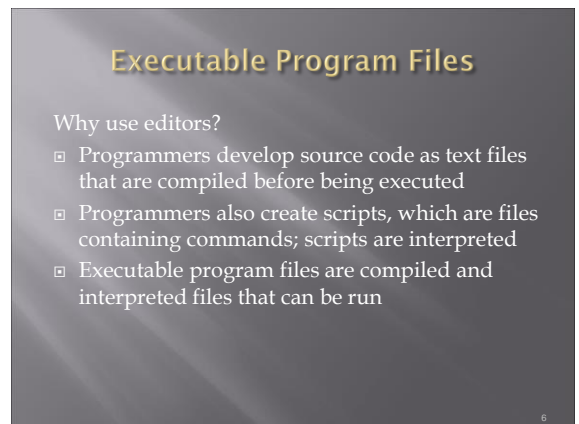
4

Printing Characters (Punctuation-Characters)				Printing Characters (Alphabetic-Uppercase)				Printing Characters (Alphabetic-Lowercase)			
Dec	Octal	Hex	ASCII	Dec	Octal	Hex	ASCII	Dec	Octal	Hex	ASCII
32	080	20	Space	65	101	41	A	97	141	61	a
33	041	21	!	66	102	42	B	98	142	62	b
34	042	22	"	67	103	43	C	99	143	63	c
35	043	23	#	68	104	44	D	100	144	64	d
36	044	24	\$	69	105	45	E	101	145	65	e
37	045	25	%	70	106	46	F	102	146	66	f
38	046	26	&	71	107	47	G	103	147	67	g
39	047	27	'	72	110	48	H	104	150	68	h
40	050	28	(73	111	49	I	105	151	69	i
41	041	29)	74	112	4A	J	106	152	6A	j
42	052	2A	*	75	113	4B	K	107	153	6B	k
43	053	2B	+	76	114	4C	L	108	154	6C	l
44	054	2C	,	77	115	4D	M	109	155	6D	m
45	055	2D	-	78	116	4E	N	110	156	6E	n
46	056	2E	.	79	117	4F	O	111	157	6F	o
47	057	2F	/	80	120	50	@	112	160	70	p
				81	121	51	A	113	161	71	q
				82	122	52	B	114	162	72	r
				83	123	53	S	115	163	73	s
				84	124	54	T	116	164	74	t
				85	125	55	U	117	165	75	u
				86	126	56	V	118	166	76	v
				87	127	57	W	119	167	77	w
				88	130	5A	X	120	170	7A	x
				89	131	5B	Y	121	171	7B	y
				90	132	5C	Z	122	172	7C	z
				96	070	38	8				
				97	071	39	9				

Special Characters (Print)				Nonprinting Characters (Abridged)			
Dec	Octal	Hex	ASCII	Dec	Octal	Hex	ASCII
58	072	3A	:	0	000	00	~(Null)
60	074	3C	<	7	007	07	Bell
61	075	3D	=	8	010	08	Backspace
62	076	3E	>	9	011	09	Tab
63	077	3F	?	10	012	0A	Line Feed, Newline
64	080	40	@	11	013	0B	Vertical Tab
				12	014	0C	Form feed
				13	015	0D	Carriage return

Figure 3-1 ASCII characters

5



6

Using Editors

- ❑ Editors let you create and edit ASCII files
- ❑ UNIX/Linux normally include a couple different editors: pico, vi and Emacs
- ❑ Pico, vi, and Emacs are screen editors: they display the text you are editing one screen at a time
- ❑ Other editors also exist (e.g. ed, ex, vim, etc.)

7

Using Nano

- ❑ To start nano, you type: nano <filename> at the command prompt.
- ❑ You will then be presented with a simple editor that will allow you to create and edit text files.

```

GNU nano 2.9.0 testfile Modified
This is a test file.
Nano is great little editor that makes it easy to create text files.
The commands at the bottom use the Ctrl key sequence to perform editing tasks.

^G Get Help  ^O Write Out  ^W Where Is  ^C Cut Text   ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^S Replace    ^U Uncut Text ^_ To Spell  ^_ Go To Line
  
```

8

Using Nano Help

```

Main nano help text
The nano editor is designed to emulate the functionality and ease-of-use of the UW Pico text editor. There are four main sections of the editor. The top line shows the program version, the current filename being edited, and whether or not the file has been modified. Next is the main editor window showing the file being edited. The status line is the third line from the bottom and shows important messages. The bottom two lines show the most commonly used shortcuts in the editor.

Shortcuts are written as follows: Control-key sequences are notated with a '^' and can be entered either by using the Ctrl key or pressing the Esc key twice. Meta-key sequences are notated with 'M-' and can be entered using either the Alt, Cmd, or Esc key, depending on your keyboard setup. Also, pressing Esc twice and then typing a three-digit decimal number from 000 to 255 will enter the character with the corresponding value. The following keystrokes are available in the main editor window. Alternative keys are shown in parentheses:

^G (F1) Display this help text
^X (F2) Close the current buffer / Exit from nano

^R Refresh  ^W Where Is  ^P Prev Line  ^N Prev Page  ^_ First Line
^C Close    ^M WhereIs Next ^_ Next Line  ^V Next Page  ^_ Last Line
  
```

9

Using Nano Help (cont.)

```

Main nano help text
^G (F1) Display this help text
^X (F2) Close the current buffer / Exit from nano
^O (F3) Write the current buffer (or the marked region) to disk
^B (F3) Insert another file into current buffer (or into new buffer)

^W (F6) Search forward for a string or a regular expression
^_ (M-R) Replace a string or a regular expression
^X (F9) Cut current line (or marked region) and store it in outbuffer
^U (F10) Uncut from the cutbuffer into the current line

^J (F4) Justify the current paragraph
^T (F12) Invoke the spell checker, if available
Invoke the linter, if available
Invoke formatter, if available

^C (F11) Display the position of the cursor
^_ (M-G) Go to line and column number

M-U Undo the last operation

^R Refresh  ^W Where Is  ^P Prev Line  ^N Prev Page  ^_ First Line
^C Close    ^M WhereIs Next ^_ Next Line  ^V Next Page  ^_ Last Line
  
```

10

Using Nano Help (cont.)

```

Main nano help text
M-U Undo the last operation
M-E Redo the last undone operation

M-A (^6) Mark text starting from the cursor position
M-G (M-^) Copy current line (or marked region) and store it in outbuffer

M-] Go to the matching bracket

M-W (F16) Repeat the last search
M-A Search next occurrence backward
M-V Search next occurrence forward

^B (C) Go back one character
^F (C) Go forward one character
^_ (M-Space) Go back one word
^_ (^Space) Go forward one word
^A (Home) Go to beginning of current line
^E (End) Go to end of current line

^R Refresh  ^W Where Is  ^P Prev Line  ^N Prev Page  ^_ First Line
^C Close    ^M WhereIs Next ^_ Next Line  ^V Next Page  ^_ Last Line
  
```

11

Using Nano Help (cont.)

```

Main nano help text
^A (Home) Go to beginning of current line
^E (End) Go to end of current line

^P (A) Go to previous line
^N (V) Go to next line
M-- (M-) Scroll up one line without moving the cursor textually
M++ (M=) Scroll down one line without moving the cursor textually

^A (M-7) Go to previous block of text
^V (M-8) Go to next block of text
M-( (M-9) Go to beginning of paragraph; then of previous paragraph
M-) (M-0) Go to just beyond end of paragraph; then of next paragraph

^Y (F7) Go one screenful up
^V (F8) Go one screenful down
M- (M=) Go to the first line of the file
M-/ (^End) Go to the last line of the file

M- (M-<) Switch to the previous file buffer

^R Refresh  ^W Where Is  ^P Prev Line  ^N Prev Page  ^_ First Line
^C Close    ^M WhereIs Next ^_ Next Line  ^V Next Page  ^_ Last Line
  
```

12

Using Nano Help (cont.)

```

Main nano help text

M-| (M-<) Switch to the previous file buffer
M-| (M->) Switch to the next file buffer

^I (Tab) Insert a tab at the cursor position
^M (Enter) Insert a newline at the cursor position

^D (Del) Delete the character under the cursor
^H (Bsp) Delete the character to the left of the cursor
Cut backward from cursor to word start
Cut forward from cursor to next word start
Cut from the cursor position to the end of the file

M-J Justify the entire file
M-D Count the number of words, lines, and characters
M-V Insert the next keystroke verbatim

^L Refresh (redraw) the current screen
^Z Suspend the editor (if suspension is enabled)

^R Refresh ^W Where Is ^P Prev Line ^N Prev Page ^O First Line
^X Close ^M WhereIs Next ^K Next Line ^V Next Page ^C Last Line
  
```

13

Using Nano Help (cont.)

```

Main nano help text

^L Refresh (redraw) the current screen
^Z Suspend the editor (if suspension is enabled)

M-| (Tab) Indent the current line (or marked lines)
M-| (Sh-Tab) Unindent the current line (or marked lines)

M-3 Comment/uncomment the current line (or marked lines)
^] Try and complete the current word

M-: Start/stop recording a macro
M-: Run the last recorded macro

^Q Search backward for a string or a regular expression

^S Save file without prompting

M-X Help mode enable/disable
M-C Constant cursor position display enable/disable
M-O Use of one more line for editing enable/disable

^R Refresh ^W Where Is ^P Prev Line ^N Prev Page ^O First Line
^X Close ^M WhereIs Next ^K Next Line ^V Next Page ^C Last Line
  
```

14

Using Nano Help (cont.)

```

Main nano help text

M-C Constant cursor position display enable/disable
M-O Use of one more line for editing enable/disable
M-S Smooth scrolling enable/disable
M-W Soft wrapping of overlong lines enable/disable
M-# Line numbering enable/disable
M-P Whitespace display enable/disable
M-T Color syntax highlighting enable/disable

M-H Smart home key enable/disable
M-I Auto indent enable/disable
M-K Cut to end enable/disable
M-L Hard wrapping of overlong lines enable/disable
M-Q Conversion of typed tabs to spaces enable/disable

M-B Backup files enable/disable
M-F Reading file into separate buffer enable/disable
M-M Mouse support enable/disable
M-N No conversion from DOS/Mac format enable/disable
M-Z Suspension enable/disable

^R Refresh ^W Where Is ^P Prev Line ^N Prev Page ^O First Line
^X Close ^M WhereIs Next ^K Next Line ^V Next Page ^C Last Line
  
```

15

Using the vi (vee-eye) Editor

- Called vi because it is visual; it immediately displays on screen the changes that you make to text
- Works in three modes
 - Insert: lets you enter text
 - Command: lets you enter editing commands
 - Extended (ex) command set: lets you use an extended set of editing commands

16

Using the vi Editor - Creating/Editing Files



To create a new file in the vi editor, type vi and the name of the new file at the command prompt

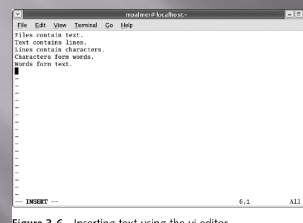
Figure 3-2 Creating a new file in the vi editor

When started, the vi editor is in command mode

- To insert text into a file, you must switch to insert mode
- You can repeat the line just entered with the repeat command (.)
- To edit what you've just typed, move the cursor with the various keyboard cursor movement keys

17

Using the vi Editor - Inserting Text



- In order to insert text, you issue the "i" command to enter insert mode

Figure 3-6 Inserting text using the vi editor

18

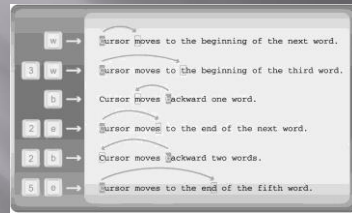
Using the vi Editor – Cursor control

Table 3-1 vi editor's cursor movement keys

Key	Movement
<i>h</i> or left arrow	Left one character position
<i>l</i> or right arrow	Right one character position
<i>k</i> or up arrow	Up one line
<i>j</i> or down arrow	Down one line
<i>H</i>	Upper-left corner of the screen
<i>L</i>	Last line on the screen
<i>G</i>	Beginning of the last line
<i>nG</i>	The line specified by a number, <i>n</i>
<i>W</i>	Forward one word
<i>b</i>	Back one word
<i>O</i> (zero)	Beginning of the current line
<i>S</i>	End of the current line
<i>Ctrl+u</i>	Up one-half screen
<i>Ctrl+d</i>	Down one-half screen
<i>Ctrl+f</i> or Page Down	Forward one screen
<i>Ctrl+b</i> or Page Up	Back one screen

19

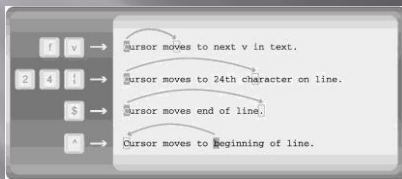
Using the vi Editor – Navigating Through a File



Moving the Cursor in vi

20

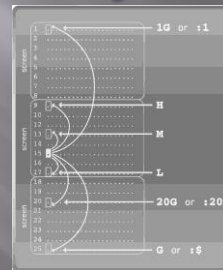
Using the vi Editor – Navigating Through a File



Moving to a Specific Textual Character in vi

21

Using the vi Editor – Navigating Through a File



Moving the Cursor by Line Address and Location Display

22

Using the vi Editor – Deleting/Searching

- While still in command mode:
 - To delete text, move to a character and then type "x"
 - You can undo a command (reverse its effects) by typing "u"
 - To search for a text pattern, type a forward slash (/), type the pattern, and press Enter

23

Using the vi Editor – Moving and Copying Text

OBJECT	OPERATIONS		
Whole line	Delete dd	Change cc	Yank yy
Rest of line	D or d\$	C or c\$	y\$
To a character <i>x</i> on the line	dfx	cfx	yfx
Word	dw	cw	yw
Character	x or dl	s or cl	yl

Commands to Move and Copy Text

24

Using the vi Editor - Delete Cmds.

Table 3-2 vi editor's delete commands

Command	Purpose
x	Delete the character at the cursor.
dd	Delete the current line (putting it in a buffer so it can also be pasted back into the file).
dw	Delete the word starting at the cursor. If the cursor is in the middle of the word, delete from the cursor to the end of the word.
d\$	Delete from the cursor to the end of the line.
d0	Delete from the cursor to the start of the line.

25

25

Using the vi Editor - Moving and Copying Text

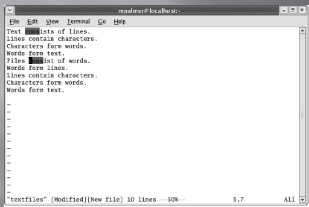
Operators	Objects	Meaning
	w	Delete, changes, or yanks the rest of the current word.
	3w	Delete, changes, or yanks three words right from cursor.
	3e	Delete, changes, or yanks right from cursor to third end of word.
	G	Delete, changes, or yanks all lines from current line to end of file.
	=	Delete, changes, or yanks all characters from cursor to beginning of line.
	\$	Delete, changes, or yanks all lines from cursor to end of line.
d (delete) c (change) y (yank)	5l	Delete, changes, or yanks all lines from cursor to line 5 (or any specified line).
	5c	Delete, changes, or yanks all characters on line from cursor forward to e.
	5x	Delete, changes, or yanks all characters on line from cursor backward to e.
	12l (pipe)	Delete, changes, or yanks all characters on line from cursor to the 12th character on the line.
	3h	Delete, changes, or yanks three characters to left of cursor.
	3H	Delete, changes, or yanks character under cursor and two more to right of cursor.
	3k	Delete, changes, or yanks line under cursor and two more above cursor.

The delete, change, and yank commands

For Examples:
 =5G change all lines from the current line to line 5.
 3x2 yank all characters from cursor to character 12 on the line.
 3H delete all characters from cursor to 8 characters to the left.

26

Using the vi Editor - Status Line



The status line at the bottom of the screen displays information, including line-oriented commands and error messages

Figure 3-11 vi status line appears at the bottom of the screen

27

27

Using the vi Editor - Searching/Replacing Text

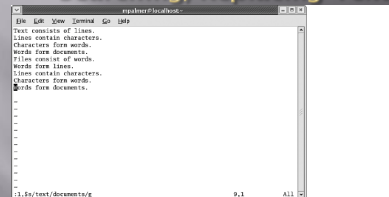


Figure 3-12 Searching for and replacing text using a line-oriented edit command

Searching and replacing is a line-oriented command that executes independently of the cursor position

28

28

Using the vi Editor - Saving/Exiting

- Saving a file and exiting vi
 - You should always save the file before exiting vi, otherwise changes are lost
 - To save a file and continue working on it, type the :w (write) command
 - While in command mode, use the :wq (write and quit) command to save and exit vi, or the :zz command to exit after saving
 - You can also use :x to save and exit

29

29

Using the vi Editor - Saving Data



Figure 3-3 Saving without exiting

30

30

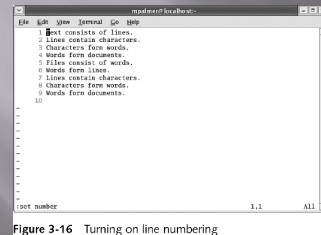
Using the vi Editor – Other Options

- ▣ In vi, you can also:
 - Add text from another file
 - Leave vi temporarily to perform other UNIX/Linux tasks, then return to your file
 - Change your display while editing, such as adding line numbering
 - Copy, cut, and paste text to help editing
 - Print text files
 - Cancel an editing session
 - Get help

31

31

Using the vi Editor – Line Numbers



```

1 1.txt consists of lines.
2 Lines contain characters.
3 Characters form words.
4 Words form documents.
5 Files consist of words.
6 Words form lines.
7 Lines contain characters.
8 Characters form words.
9 Words form documents.
10

```

Turn on line numbering when you want to work with a range of lines and refer to the line numbers to specify text

Figure 3-16 Turning on line numbering.

32

32

Unit Summary

- ▣ Bytes are computer characters stored using numeric code (e.g., ASCII)
- ▣ The Nano editor is a simple, easy-to-use editor that supports file creation and text editing.
- ▣ The vi editor is a popular choice among UNIX/Linux users to edit text files

33

33