Sheridan

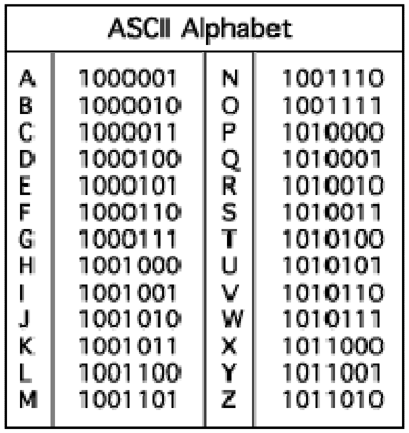
SYST13416
Introduction to Linux
Operating

Text Editing

ASCII TEXT FILES

Objectives

- Understand, recognize, and view various types of files
- Understand what is ASCII code and how it is used by OS
- Command: **file** – determine file type.
- Command: **strings** - print the strings of printable characters in files.
- Command: **od** - dump files in octal and other formats.
- Create text files with **touch** command
- Create text files using a Line Editor (**cat >**)
- Display text files with **cat** command
- Display text files with **head** and **tail** commands
- Display text files with **more** and **less** commands
- Modify and edit text files with full text editor (**vi**)



ASCII Alphabet			
A	1000001	N	1001110
B	1000010	O	1001111
C	1000011	P	1010000
D	1000100	Q	1010001
E	1000101	R	1010010
F	1000110	S	1010011
G	1000111	T	1010100
H	1001000	U	1010101
I	1001001	V	1010110
J	1001010	W	1010111
K	1001011	X	1011000
L	1001100	Y	1011001
M	1001101	Z	1011010

The ASCII Code

- Before the 1990’s, the character encoding used by most computers was ASCII code, simply referred to as ASCII (*pronounced “ask-key”).* The name stands for “American Standard Code for Information Interchange”. It was created in 1967, specifying a 7-bit pattern for every character, from 000 0000 to 111 1111.
- The 128 characters that comprise the ASCII code consist of 33 control characters and 95 printable characters (52 letters of the alphabet, 10 numbers, 32 punctuation symbols, and the space character):

```
   !”#$%&’()*+,-./0123456789:;<=>?
  @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
  `abcdefghijklmnopqrstuvwxyz{|}~
```

- Gradually additional characters were added and the Extended ASCII code comprises 256 characters (8-bit patterns); for tables, see
  http://www.asciitable.com/
  http://www.pcguide.com/res/tablesASCII-c.html

- Today, it is a **standard method for translating binary numbers into alphanumerical representation**; a standardized set of 8-bit patterns.
- Example: lower case a = 01100001 binary = 97 decimal
- On Linux systems, you can check the ASCII code with the command:
  `man ascii`

Non-text Files

- Graphic information, such as images, can include strings of bits representing white and black dots, where each black dot represents a 1 and white dot a 0
- Graphics files - bit patterns--rows and columns of dots called a bitmap—translated by graphics software. It transforms a complex array of bits into an infinite variety of images.
- Colors are represented by a group of bits for each dot (pixel).

NAME
file – identifies files by examining their contents.

SYNOPSIS
`file [OPTION] [FILE]`

DESCRIPTION
See Manual Page.

OPTIONS
See Manual Page.

NAME
strings – displays readable text in files.

SYNOPSIS
`strings [OPTION] [FILE]`

DESCRIPTION
See Manual Page.

OPTIONS
See Manual Page.

file command

- Check the file type of .login file in the Unix hierarchy (It is a special kind of text file)
  ```
  file .login
  ```
- Try this command on the various files you find on the Unix machine.
- The "file" command tries to identify files by examining their contents. Providing a name of an image file as argument will produce similar output to the following:
  ```
  file tux_small.png
  tux_small.png: PNG image data, 128 x 151, 8-bit/color RGB, non-interlaced
  ```

strings Command

- It is important to employ text file commands, such as cat, tac, head, tail, more, and less, only on files that contain text; otherwise, you might find yourself with random output on the terminal screen or even a dysfunctional terminal.
- To view the contents of binary files, you typically use the program that was used to create the file; however, some commands can be used to safely display the contents of most binary files. The strings command searches for text characters in a binary file and outputs them to the screen. In many cases, these text characters might indicate what the binary file is used for. For example, to find the text characters inside the /bin/echo binary executable program page-by-page, you could use the following command:
  ```
  strings /bin/echo | more
  ```
- The output will start something like this:
  ```
  /lib/ld-linux.so.2
  PTRh|
  <nt7<e
  |[^_]
  [^_]
  [^_]
  Try `%s --help' for more information.
  Usage: %s [OPTION]... [STRING]...
  --More--
  ```

od Command

- Although this output might not be easy to read, it does contain portions of text that can point a user in the right direction to find out more about the /bin/echo command. Another command that is safe to use on binary files and text files is the od command, which displays the contents of the file in octal format (numeric base 8 format). An example of using the od command to display the first five lines of the file project4 is shown in the following example:

  `od project4 | head -5`

```
0000000 064510 072040 062550 062562 020054 020111 067550 062560
0000020 072040 064550 020163 060544 020171 064546 062156 020163
0000040 067571 020165 062567 066154 006456 006412 052412 063156
0000060 071157 072564 060556 062564 074554 073440 020145 062567
0000100 062562 067040 072157 060440 066142 020145 067564 066440
--More--
```

Note

You can use the -x option with the od command to display a file in hexadecimal format (numeric base 16 format).

Interesting Links:

http://www.nickciske.com/tools/binary.php

NAME

od - display a file in octal format

SYNOPSIS

`od [OPTION] [FILE]`

DESCRIPTION

The od command displays the contents of FILEs to standard output. See Manual Page.

OPTIONS

-x	display a file in hexadecimal format

Display files

- **more**: page through file
- **less**: page through file
- **head**: display beginning of a file
- **tail**: display end of a file
- **cat**: display entire file(s)

Create files

- **touch**: create empty file
- **touch**: change the last access time and last modification time of a file
- **command > file:** by redirection
- **vi**: full text editor

Text Files

Create Text files with touch command

First, make sure you are in your lab4 sub-directory. There are different ways you can create files. Use the command touch to create an empty file:

```
touch file1
touch –m –t 01012010 file1
```

Create Text Files using a Line Editor

Use the command cat with redirection (operator >):

```
cat > file2
This is the first line of file2
This is the second line of file2
```

At the end of each line press enter. Note that you are inside a line editor—you can change only the current line but you cannot return to a previous line. To return back to the prompt, use the key combination CTRL+D.

Create a file using redirection:

```
man man > file3
```

Display the contents of file3.

```
cat file3
```

Note the file is very long, most of it flashes before you and you can view only the last part of it. To view large files a section at a time, it is more practical to use the commands more, less, head, and tail.

Display Text Files: con**cat**enate

You have already been using the cat command to display the content of a text file. Try the following, noting each output:

```
cat file3
cat file3 file2
cat file2 file3
cat /etc/passwd
```

Examine the manual page for the cat command, read the description and browse through the available options.

Note: if you need to interrupt the output, use the key combination Ctrl+C

Display Text Files: heads or tails?

To display the beginning of a file, use the command head.

```
head file3
head /etc/passwd
```

To display the beginning of a file, use the command tail.

```
tail file3
tail /etc/passwd
```

By default, the command displays 10 lines. To specify the number of lines you wish to display, use the –N option, where N is replaced by the number of lines to display:

```
head -25 file3
tail -100 /etc/passwd
```

Examine the manual pages for the commands **head** and **tail**, read the description and browse through the available options.

Display Text Files: more or less…

Change back to your lab4 subdirectory. Browse through the manual pages for the commands **more**, **less**, and **pg**. Read the description of each and examine the available options. If you cannot find a manual page, try an Internet search. Note similarities and differences among **more**, **less**, and **pg**.

Display file3 using the following commands, noting the output:

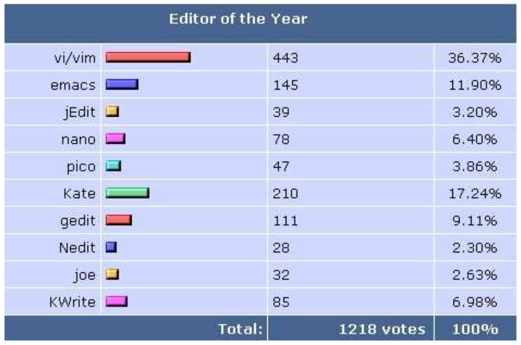
```
more file3
```

Try moving using the keys enter, spacebar, b, and q

```
less file3
```

Try moving using the keys enter, spacebar, b, and q

- The **less** command offers a number of advantages over **more**:
  - You can scroll backwards and forwards through text files using your cursor keys.
  - You can navigate through files with bookmarks, by line numbers, or by percentage of file.
  - Sophisticated searches, pattern options, and highlighting through multiple files.
  - Compatible keystrokes with word processing programs, such as emacs.
  - The less command won't quit on you when you reach the end of a file, or the end of the standard input.
  - The information prompt at the bottom of the screen is more customizable and offers more information.
  - Loads of options, including a separate key setup program, lesskey, so you can customize which keys control less.



Editor of the Year			
vi/vim	🟥	443	36.37%
emacs	🟦	145	11.90%
jEdit	🟨	39	3.20%
nano	🟪	78	6.40%
pico	🟦	47	3.86%
Kate	🟩	210	17.24%
gedit	🟥	111	9.11%
Nedit	🟦	28	2.30%
joe	🟨	32	2.63%
KWrite	🟪	85	6.98%
	Total:	1218 votes	100%

Why operating system editors use ASCII files

- Operating system editors let you create and edit simple ASCII files that you can read and understand. They interpret the human-readable data and convert it into machine-understandable code. UNIX includes some editors, which may include
    - **vi** (can be found on every installation and every distribution of Unix/Linux)
    - **emacs**
    - **pico**
- Also called screen editors, they display the text you are editing on screen and let you move around the screen to change and add text.

Full text Editing: using vi editor

- vi is visual: immediately displays on screen the changes you make to text
- vi is modal: works in two modes
- The command for editing the file "filename" is

```
vi filename
```

Text deletion commands

x	Deletes character under cursor
dw	Deletes from cursor to beginning of next word
dd	Deletes line containing cursor

Commands can be preceded by an number to indicate the number of characters, words, or lines to be affected.

Text alternation commands

r	Replace character under cursor with next char typed
cw	change word (beginning at cursor) to new text
J	join next line down to line with cursor
u	undo last command

Text moving commands

yy	yank a copy of a line, place it in a buffer
p	put after the cursor the last item yanked or deleted
P	put before the cursor the last item yanked or deleted

Saving text and quitting

:w	write the current text into the permanent file
:q	quit, if no changes since last write
:q!	quit, discarding all changes since last write
:wq	save and quit

Command Mode

- Lets you use the available commands
- You are placed in a Command Mode when you invoke vi
- To enter the Command Mode from the Text Mode, hit the <esc> key.

Text Mode

- Lets you use the keyboard to enter text
- Any of the following commands will put you in the Text Mode: a, I, o, O, R, and c

i	Insert before cursor
I	Insert at the beginning of current line
a	Append text after cursor position
A	Append at the end of the current line
o	Open a new line before the cursor position
O	Open a new line after the cursor position

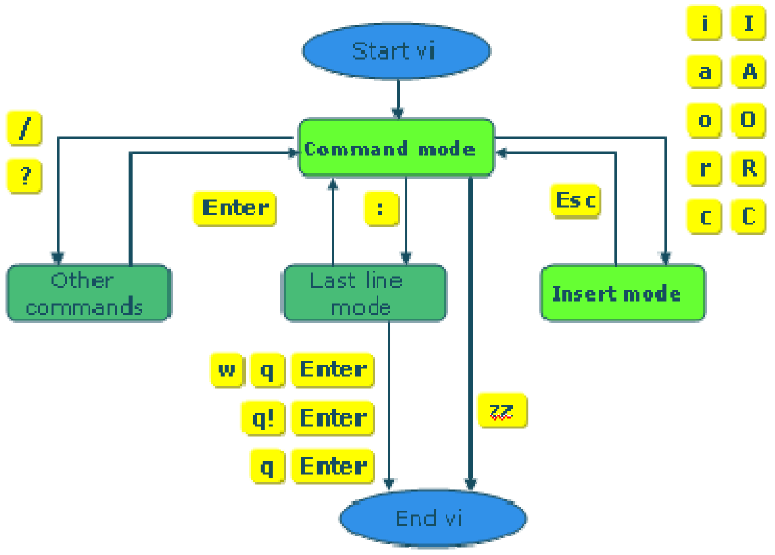


Table of Contents