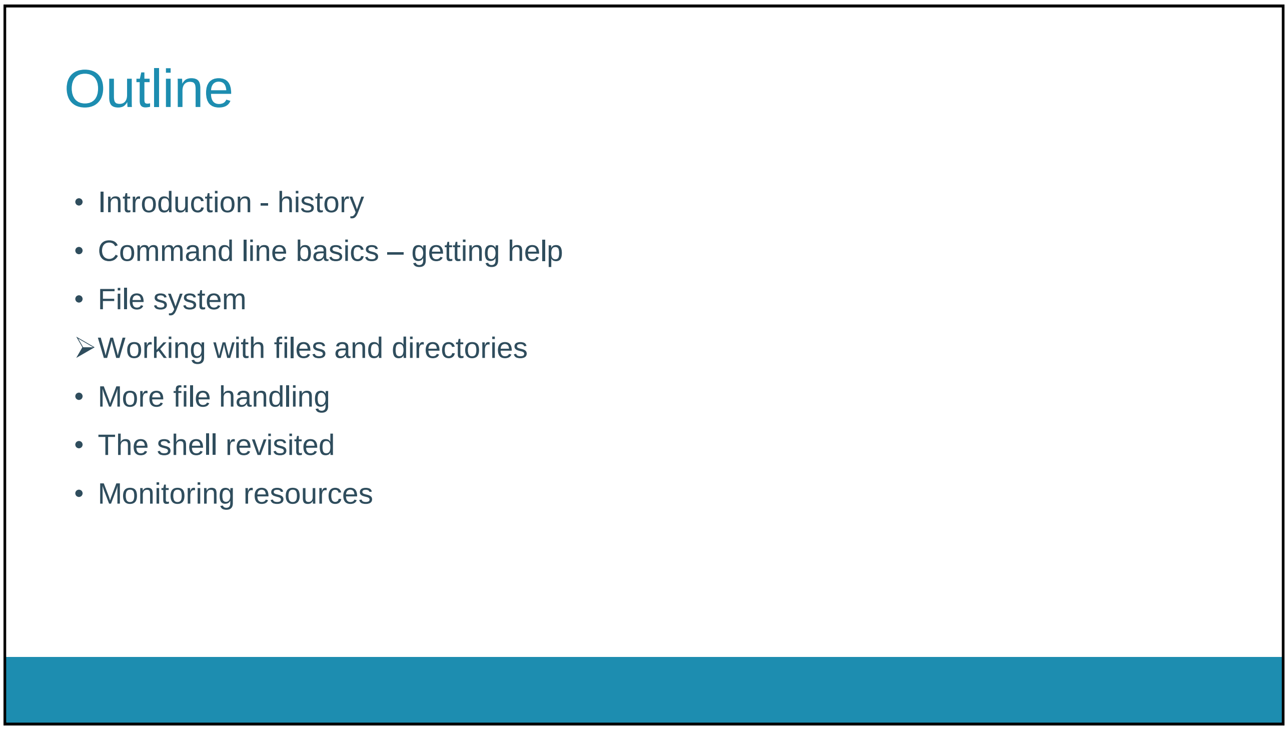
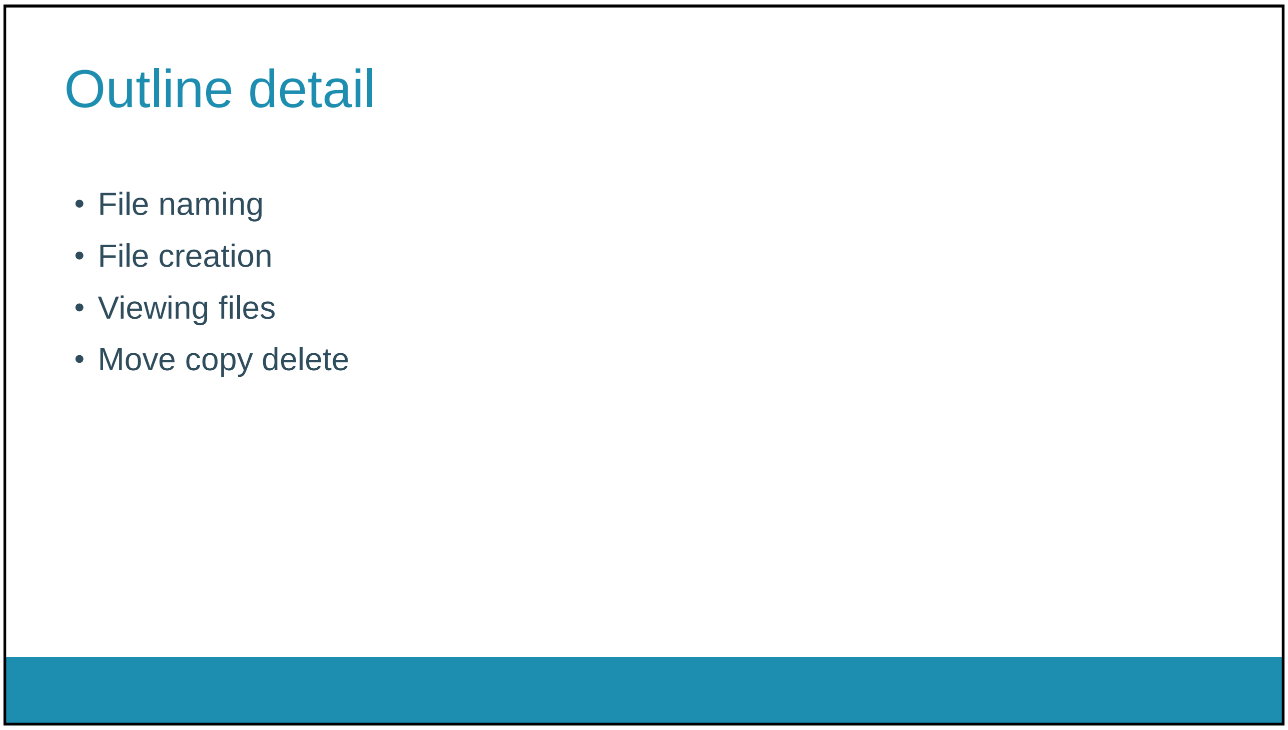


Outline

- Introduction - history
- Command line basics – getting help
- File system
- ➢Working with files and directories
- More file handling
- The shell revisited
- Monitoring resources

1



Outline detail

- File naming
- File creation
- Viewing files
- Move copy delete

2

Working with Files and Directories

3

Naming

- **Do not use spaces.**
  You can use - or _ instead
- **Do not begin the name with - (dash).**
  Commands treat names starting with - as options.
- **Stick with letters, numbers, . (dot), - (dash) and _ (underscore).**
  Many other characters have special meanings on the command line.
- **Meaningful name.**
- **File extension or not.**
  Is just a convention, in Linux file extensions are not necessary. Files contain bytes. However, two-part names are used to help keep different kinds of files apart.

4

Creating files: text editing

5

Primitive text editing

- Combine Redirection and Viewing
- use `cat` to direct stdin to a text file
  - `$ cat > my_text.txt`
  - Enter text.
  - To end the text input, press **Ctrl-D**.
- Check with `cat my_text.txt`
- Try adding another line of text to the existing file
  - `$ cat >> my_text.txt`

6

Primitive text editing

- Create an empty file
  - `>filename`: Use redirection to create an empty file filename.
  - `touch filename`: create an empty file, if the file does not exist yet.

7

echo command

- `echo`: displays line of text/string that are passed as an argument .
  - built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.
- Syntax: `echo [options] [input string]`
- 2 kinds of quoting: weak (") and strong (‘).
  - Use strong quoting (‘): nothing is interpreted,
  - Use weak quoting (“): variable expansion, command expansion works

```
echo "Path to your shell is: $SHELL, $(ls)"
echo 'Path to your shell is: $SHELL, $(ls)'
```

8

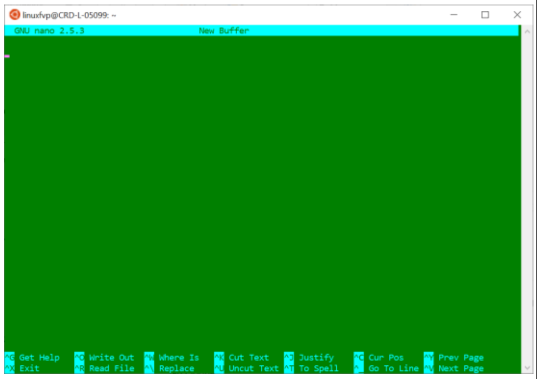
Text editor

- Tool to create and edit files.
- There is no best text editor; it depends on personal taste.
- Text-only text editors
  - Simplicity first:
    - nano
  - With a steep learning curve: (needed for sysadmins and great for power users)
    - vi, vim
    - emacs
- Graphical text editors
  - Gedit: general purpose GUI based text editor

9

nano

- Entering text: nano is a "modeless" editor. This means that all keystrokes, with the exception of Control and Meta sequences, enter text into the file being edited.
- Commands (lower part) are given by using the Control key (Ctrl, shown as ^) or the Meta key (Alt or Cmd, shown as M-).
  - A control-key sequence is entered by holding down the Ctrl key and pressing the desired key.
  - A meta-key sequence is entered by holding down the Meta key (normally the Alt key) and pressing the desired key.
- Manual: https://www.nano-editor.org/dist/v4/nano.pdf
- Cheat sheet: https://www.nano-editor.org/dist/latest/cheatsheet.html



```
linuxfvp@CRD-L-05099: ~
  GNU nano 2.5.3                   New Buffer

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    ^Y Prev Page
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line ^V Next Page
```

10

vi(m)

- Text-mode text editor available in all Linux systems.
- Created before computers with mice appeared.
- Very productive for power users.
- Check the web for tutorials:
  - https://upload.wikimedia.org/wikipedia/commons/d/d2/Learning_the_vi_Editor.pdf
  - ftp://ftp.vim.org/pub/vim/doc/book/vimbook-OPL.pdf

11

vi(m)

- 2 basic modes of operation:
  - *command mode* and *editing mode*.
  - Command Mode: signals from the terminal are interpreted as editing commands.
  - Editing mode: letters typed at the keyboard are inserted into the editing buffer.
- Pressing **Esc** on the keyboard activates command mode.

Key(s)	Function	Key(s)	Function
:w	Save	A	Append text after
:x	Save and exit	r	Replace text before cursor
:q	Quit	R	Replace text after cursor
I	Insert text after	i	Insert text before
P	Paste copied text	yy	Copy current line
a	Append text before	/[TEXT]	Search for the specified text

12

vi(m)

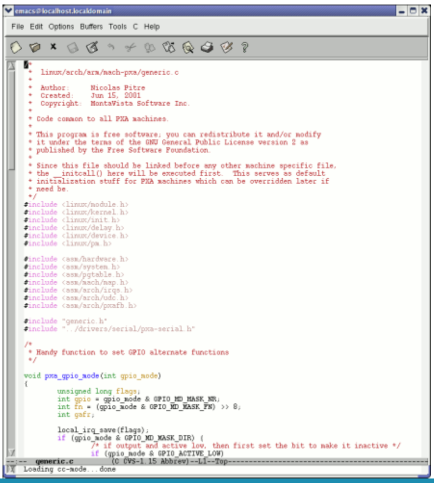
- 2 modes
  - Editing(Input) mode
    - ESC to back to cmd mode
  - Command mode
    - Cursor movement
      - h (left), j (down), k (up), l (right)
      - ^f (page down)
      - ^b (page up)
      - ^ (first char.)
      - $ (last char.)
      - G (bottom page)
      - :1 (goto first line)
    - Switch to input mode
      - a (append)
      - i (insert)
      - o (insert line after
      - O (insert line before)

- Delete
  - dd (delete a line)
  - d10d (delete 10 lines)
  - d$ (delete till end of line)
  - dG (delete till end of file)
  - x (current char.)
- Paste
  - p (paste after)
  - P (paste before)
- Undo
  - u
- Search
  - /
- Save/Quit
  - :w (write)
  - :q (quit)
  - :wq (write and quit)
  - :q! (give up changes)

13

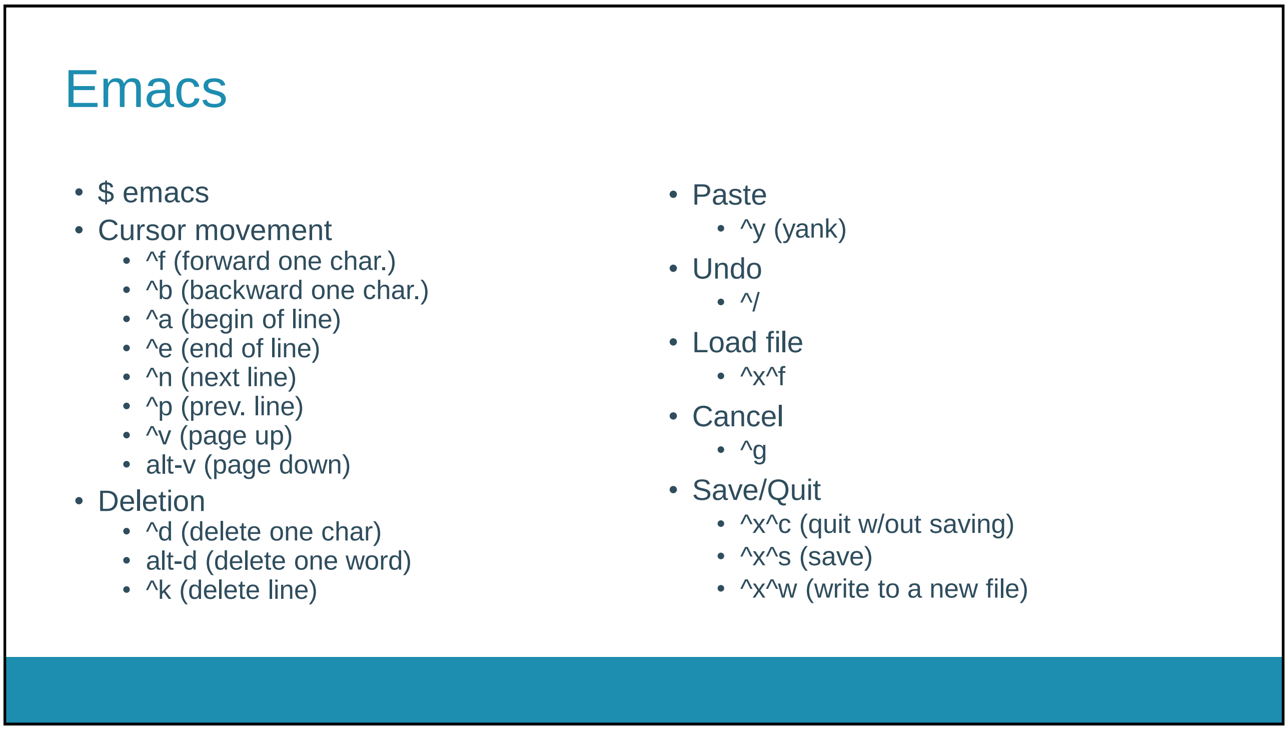
Emacs

- Extremely powerful text editor features
- Great for power users
- Non standard shortcuts
- Much more than a text editor (games, e-mail, shell, browser).
- Some power commands have to be learnt.


emacs@localhost.localdomain
File Edit Options Buffers Tools C Help

```
/*
 *  linux/arch/arm/mach-pxa/generic.c
 *
 *  Author:     Nicolas Pitre
 *  Created:    Jun 15, 2001
 *  Copyright:  MontaVista Software Inc.
 *
 * Code common to all PXA machines.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 *
 * Since this file should be linked before any other machine specific file,
 * the __initcall() here will be executed first.  This serves as default
 * initialization stuff for PXA machines which can be overridden later if
 * need be.
 */
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/delay.h>
#include <linux/device.h>
#include <linux/pm.h>

#include <asm/hardware.h>
#include <asm/system.h>
#include <asm/pgtable.h>
#include <asm/mach/map.h>
#include <asm/arch/irqs.h>
#include <asm/arch/udc.h>
#include <asm/arch/pxafb.h>

#include "generic.h"
#include "../drivers/serial/pxa-serial.h"

/*
 * Handy function to set GPIO alternate functions
 */

void pxa_gpio_mode(int gpio_mode)
{
        unsigned long flags;
        int gpio = gpio_mode & GPIO_MD_MASK_NR;
        int fn = (gpio_mode & GPIO_MD_MASK_FN) >> 8;
        int gafr;

        local_irq_save(flags);
        if (gpio_mode & GPIO_MD_MASK_DIR) {
                /* if output and active low, then first set the bit to make it inactive */
                if (gpio_mode & GPIO_ACTIVE_LOW)
--:--  generic.c        (C CVS-1.15 Abbrev)--L1--Top------------------------------------
Loading cc-mode...done
```

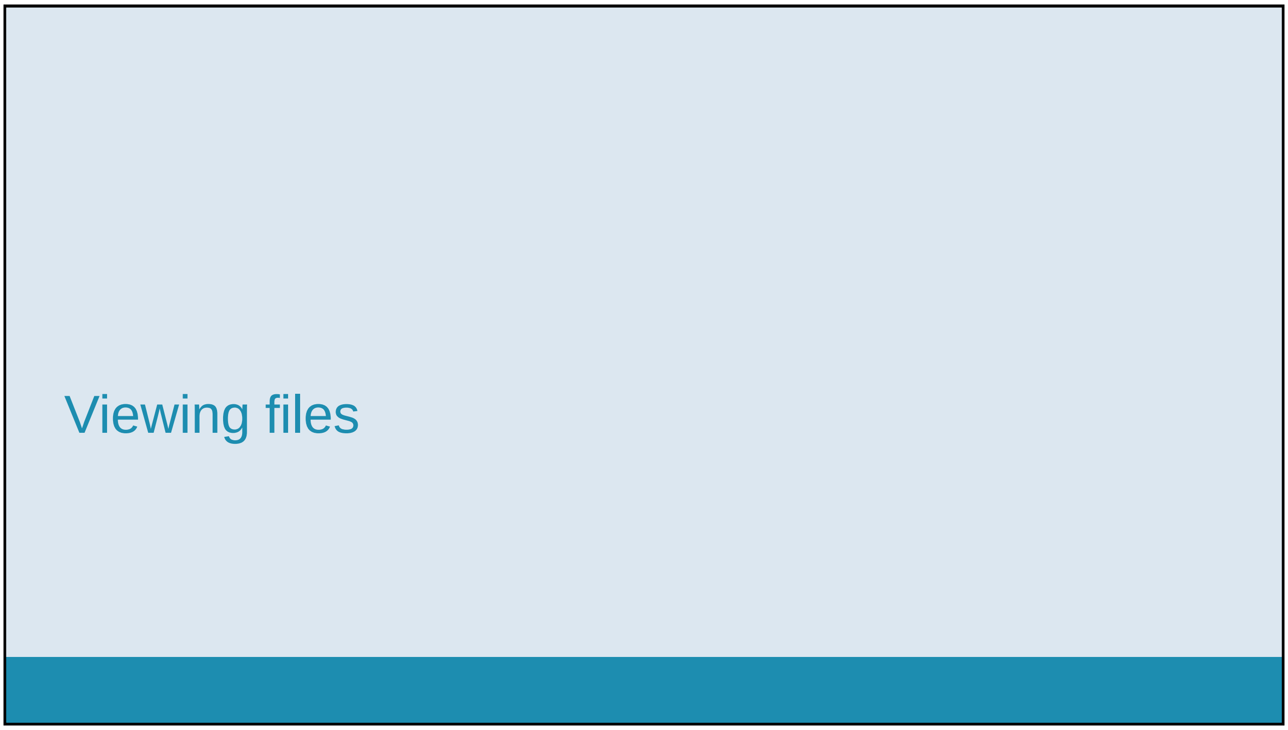
14



Emacs

- $ emacs
- Cursor movement
  - ^f (forward one char.)
  - ^b (backward one char.)
  - ^a (begin of line)
  - ^e (end of line)
  - ^n (next line)
  - ^p (prev. line)
  - ^v (page up)
  - alt-v (page down)
- Deletion
  - ^d (delete one char)
  - alt-d (delete one word)
  - ^k (delete line)
- Paste
  - ^y (yank)
- Undo
  - ^/
- Load file
  - ^x^f
- Cancel
  - ^g
- Save/Quit
  - ^x^c (quit w/out saving)
  - ^x^s (save)
  - ^x^w (write to a new file)

15



Viewing files

17

Displaying file contents

Several ways of displaying the contents of files (without editing).

- Easy solution: `cat`, prints the entire file onto the screen
  - `$ cat file1`
    displays the contents of the given file.
  - `$ cat file1 file2 file3 ...` (concatenate)
    Concatenates and outputs the contents of the given files.
- A scalable solution: one page at a time.
  - `$ more file1`
    Display the output of a command or text file one page at a time.
  - `$ less file1`

18

less

- Does more than `more`
- Does not read the whole file before starting.

Command	Description
Space bar	Advance 1 page
b	Go back 1 page
g or <	Go to the first line
G or >	Go to the last line
/<text>	Search forward for <text>
?<text>	Search backward for <text>
n	Find next match
N	Find previous match
h	Display help
q	Exit the less viewer

19

The head and tail commands

- Quick look
- `$ head [-<n>] <file>`
  Displays the first <n> lines (or 10 by default) of the given file.
  Doesn't have to open the whole file to do this!
- `$ tail [-<n>] <file>`
  Displays the last <n> lines (or 10 by default) of the given file.
  No need to load the whole file in RAM! Very useful for huge files.
- `$ tail -f <file>` (follow)
  Displays the last 10 lines of the given file and continues to display new lines when they are appended to the file.
  Very useful to follow the changes in a log file, for example.

20

File manipulation

Move Copy Delete

22

File Manipulation

- For all file manipulation commands relative of absolute paths can be used (or just files in the current directory when no extra path specified)
- Most of the commands are intuitive – shortcuts of English names

23

Directories

- Create directories
  - The `mkdir` command is used to create directories
  - `$ mkdir dir1 dir2 dir3`
- Remove directories
  - The `rmdir` command removes directories
  - `$ rmdir dir1`
  - `rmdir` will only remove empty directories.

24

Copy a file

- The `cp` command copies files and directories
- The default behavior will overwrite any existing file(s). The **-i** option overrides this behavior and prompts the user before overwriting the destination file.
- **syntax: `cp [OPTIONS] [SOURCE] [DESTINATION]`**
  - `$ cp <source_file> <target_file>`
    Copies the source file to the target.
  - `$ cp file1 file2 file3 ... dir`
    Copies the files to the target directory (last argument).
  - `$ cp -i` (interactive)
    Asks for user confirmation if the target file already exists
  - `$ cp -r <source_dir> <target_dir>` (recursive)
    Copies the whole directory.
  - `$ cp -v` (verbose)
    Displays what has been copied

25

Move or rename files

- Move or rename files and directories: `mv`
- The default behavior will overwrite any existing file(s)!
- **syntax: `mv [OPTIONS] [SOURCE] [NEW NAME/DESTINATION]`**
  - `$ mv old_name new_name`
  - If new_name is a directory name, then the file will be moved to that directory
  - `$ mv -i` (interactive)
    If the new file already exits, asks for user confirm
- The `mv` command can also be used to move or rename directories
  - `$ mv NewFiles/ OldFiles/`
  - -r option is not necessary

26

Remove files

- The rm command removes files.
- `$ rm file1 file2 file3 ...`
  Removes the given files.
- `$ rm -i` (interactive)
  Always ask for user confirmation.
- `$ rm -r dir1 dir2 dir3` (recursive)
  Removes the given directories with all their contents.
  be careful!
- Be careful using wildcards, always run `ls` first to check

27

Links

28

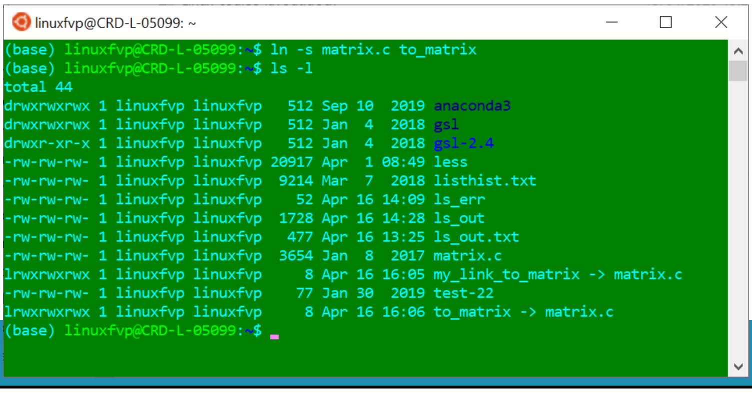
links

- Copy (`cp`) duplicates the data
  - Can be a problem with large files
  - Links create a virtual copy
- **Soft link**:
  - A.k.a. symbolic link, symlink
  - Similar to a *shortcut* in Windows.
  - Special kind of file pointing at a different file.
  - It is an indirect pointer to a file or directory; can even point to a file or a directory on a different filesystem or partition.

30

Create links

- Create with the `ln` command using the `-s` option.
  `ln -s [OPTIONS] FILE LINK`
- Check with `ls -l`
- The first character “l”, indicates that the file is a symlink.
- The “->” symbol shows the file the symlink points to.


linuxfvp@CRD-L-05099: ~

```
(base) linuxfvp@CRD-L-05099:~$ ln -s matrix.c to_matrix
(base) linuxfvp@CRD-L-05099:~$ ls -l
total 44
drwxrwxrwx 1 linuxfvp linuxfvp   512 Sep 10  2019 anaconda3
drwxrwxrwx 1 linuxfvp linuxfvp   512 Jan  4  2018 gsl
drwxr-xr-x 1 linuxfvp linuxfvp   512 Jan  4  2018 gsl-2.4
-rw-rw-rw- 1 linuxfvp linuxfvp 20917 Apr  1 08:49 less
-rw-rw-rw- 1 linuxfvp linuxfvp  9214 Mar  7  2018 listhist.txt
-rw-rw-rw- 1 linuxfvp linuxfvp    52 Apr 16 14:09 ls_err
-rw-rw-rw- 1 linuxfvp linuxfvp  1728 Apr 16 14:28 ls_out
-rw-rw-rw- 1 linuxfvp linuxfvp   477 Apr 16 13:25 ls_out.txt
-rw-rw-rw- 1 linuxfvp linuxfvp  3654 Jan  8  2017 matrix.c
lrwxrwxrwx 1 linuxfvp linuxfvp     8 Apr 16 16:05 my_link_to_matrix -> matrix.c
-rw-rw-rw- 1 linuxfvp linuxfvp    77 Jan 30  2019 test-22
lrwxrwxrwx 1 linuxfvp linuxfvp     8 Apr 16 16:06 to_matrix -> matrix.c
(base) linuxfvp@CRD-L-05099:~$
```

31

Create links

- Editing a symbolic link file is the same as editing the source file
- Deleting the symbolic link does not delete the source file.
- Deleting the source file leaves a dangling link
- `$ ln -s file_v5.doc file_final.doc`
  creates a symbolic link called file_final.doc that points to file_v5.doc
- `$ ln -s /home/demo/dir1/dir2/dir3 /home/demo/jump2dir`
  creates a symbolic link called jump2dir that points to a deep directory (allows for quicker access)

32

Hands-on

34