

Finite element methods in scientific computing

Wolfgang Bangerth, Colorado State University

Lecture 2.9:

A (very brief) introduction to Linux Part 1: The command line

On the use of the command line

When working on linux:

- You can work with the file manager
- You can work on the command line
- If you know what you do, the command line is usually faster
- You get a command line by running a *shell* inside a *terminal window*

Let us look at the most common command line operations!

On the use of the command line

Common commands:

- *ls* – list the contents of the current directory
- *ls -l* – provide a *long* listing
- *cd abc* – change current *directory* to *abc*
- *mkdir abc* – *make directory abc*
- *rmdir abc* – *remove directory abc*
- *pwd* – *print (current) working directory*
- *rm file* – *remove file file*
- *rm -r dir* – recursively remove contents of *dir*

On the use of the command line

Edit (text) files:

- *kate file*
- *kwrite file*
- *gedit file*
- *nano file*
- *mousepad file*
- ...

Not all of these programs will be available on all systems, but some almost always are.

On the use of the command line

Other useful tricks:

- “up” and “down” keys allow you to select previous commands
- “Tab” autocompletes command names and file names

On the use of the command line

Commands currently running block the command line:

- Run an editor with a file from the command line
- Try to enter another command while editor still open
- To put a command into background, use '&':
gedit file &
- Or, if you forgot when you started the program:
gedit file
Ctrl-Z (suspend currently running program)
bg (put susp. program into *background*)

“Piping” input/output between programs

Input and output for programs on the command line:

- When you run a program on the command line, it
 - reads input from the keyboard (“stdin”)
 - writes regular output to the screen (“stdout”)
 - writes error messages to the screen (“stderr”)

- Some programs may of course...
 - ...not care about any input
 - ...not write anything to the screen

“Piping” input/output between programs

Input and output for programs on the command line:

Example:

ls -l

- Does not read anything
- Writes directory listing to the screen
- May write error messages to the screen, for example for *ls -l /some/file/that/does/not/exist*

“Piping” input/output between programs

Using one program's *output* as the *input* of another:

- Very useful if the second program is a “filter”
- Example:

```
ls -l | grep vtk
```
- The '*grep*' program
 - reads every line it gets
 - outputs those lines in which 'vtk' appears
- **Result:** List all (and only) 'vtk' files.

“Piping” input/output between programs

Using one program's *output* as the *input* of another:

- Very useful if the second program is a “filter”
- Example:

```
ls -l | grep vtk | wc -l
```
- The '`wc`' program
 - reads every line it gets
 - outputs number of lines, words, characters in the input
- '`wc -l`' only outputs the number of lines
- **Result:** Show the *number* of 'vtk' files.

“Piping” input/output between programs

Using one program's *output* as the *input* of another:

- Very useful if the second program is a “filter”
- Another example:

```
cat step-1.cc | grep for | wc -l
```
- The '*cat*' program
 - reads one or more files
 - outputs them to the screen
- **Result:** Count the 'for' statements in *step-1.cc*.
(But also other occurrences of the text 'for'.)

“Piping” input/output between programs

Using one program's *output* as the *input* of another:

- Very useful if the second program is a “transformer”

- Example:

```
cat step-1.cc | sed s/tr/tria/ > step-1-mod.cc
```

- The '*sed*' program and its *s//* command
 - reads every line it gets
 - replaces text
 - outputs the rest
- '>' the redirects output to a file

“Piping” input/output between programs

Using one program's *output* as the *input* of another:

- Very useful if the second program is “interactive”

- Example:

```
cat step-1.cc | less
```

- The '*less*' program
 - reads every line it gets
 - displays one page at a time
 - allows you to scroll up or down

“man” pages

To learn more about a program:

- Every unix/linux tool has a “man” page (“manual page”)
- See it on the command line via
man grep
- Many also have web sites
- Programs definitely worth learning about:
 - grep
 - sed
 - sort
 - head/tail

Summary

About the command line:

- Seems clunky at first, if you're used to graphical user interfaces
- Requires a bit of learning...
- ...but makes you *soo* much more productive if you know the basics!

Finite element methods in scientific computing

Wolfgang Bangerth, Colorado State University