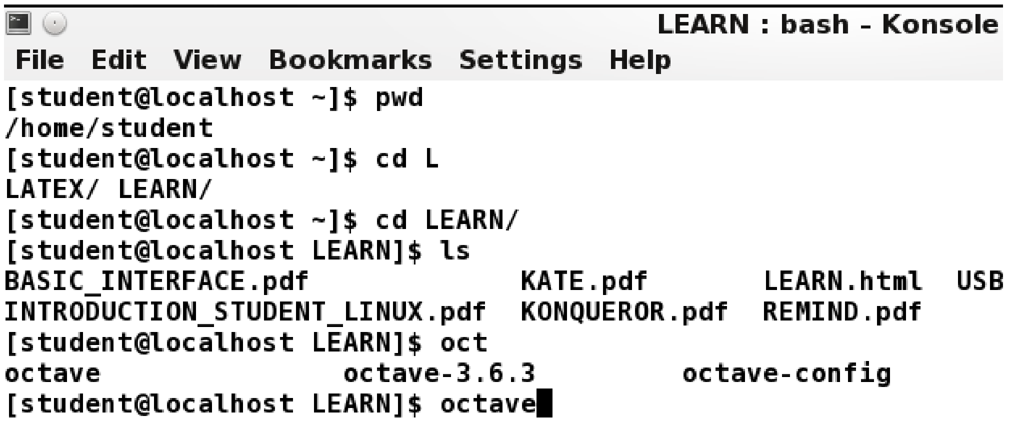
Konsole and Line commands

`Konsole` is a terminal where you pass commands directly to the system instead of working with a graphical [user] interface ("GUI").



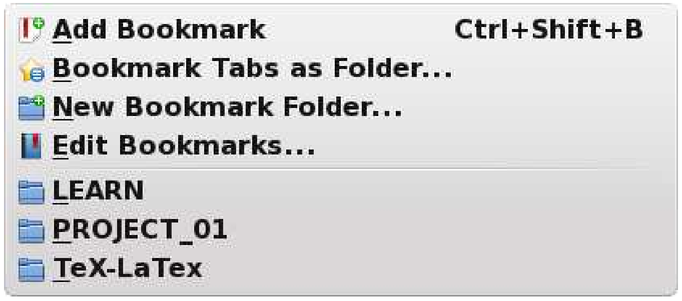
```
[student@localhost ~]$ pwd
/home/student
[student@localhost ~]$ cd L
LATEX/ LEARN/
[student@localhost ~]$ cd LEARN/
[student@localhost LEARN]$ ls
BASIC_INTERFACE.pdf             KATE.pdf        LEARN.html  USB
INTRODUCTION_STUDENT_LINUX.pdf  KONQUEROR.pdf   REMIND.pdf
[student@localhost LEARN]$ oct
octave              octave-3.6.3        octave-config
[student@localhost LEARN]$ octave
```

1. The figure shows several examples of line commands:
   i. "pwd" (= print the workind directory) + ENTER. This is very useful when you want to know where you are!
   ii. "cd" (= change directory). This example illustrates a very useful feature when working with a terminal, especially when the name is long or when you can not remember the exact name. I type: "cd L" and then TAB. The system suggests two possibilities: LEARN and LATEX. Since I want to go to LEARN, I add the letter "E" to "cd L" and push TAB again. The system now completes the word and produces: "cd LEARN". Now all I have to do is push the ENTER key and I am in the directory LEARN.

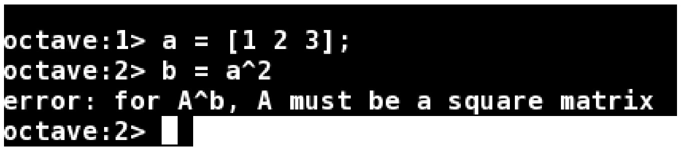
      If you are in some sub-directory and you want to go to /home/student, which is your main directory, then just type "cd" without specifying any further sub-directory.
   iii. "ls" (= list). Now that I am in LEARN, I want to see which files are there and so I type ls and then ENTER. If I just want to see which PDF files are in LEARN, I would type "ls *.pdf". Here the asterisk stands for any number of missing characters and so is called a "wild card".
   iv. In (2) and (3) we were look for a directory, but the same thing holds for commands. Try typing "o" plus TAB, them add a "c" to form "oc" plus TAB and finally—as in the snapshot—"oct" plus TAB. There are now three choices, try the simplest first by adding "ave" and then TAB.

      If you do the same with "k" you will find a list of the KDE commands which are available, plus some others. This is an interesting way to learn about "hidden" programs, e.g. "kiconfinder". If you type "kiconfinder konqueror" it will tell you where the konqueror icon is located. If you do not like it, you could become the the root user and change it. You could type "kiconfinder –help" to learn more about this command.

2. As is the case with `Konqueror` and `Kate` you can have bookmarks in `Konsole`.



3. Copying and pasting does not work the same way as in `Konqueror` and `Kate`. In the example shown, I have made a mistake while trying to take the squares of the individual elements of `a` (I forgot the dot: **.^**)



```
octave:1> a = [1 2 3];
octave:2> b = a^2
error: for A^b, A must be a square matrix
octave:2> 
```

   To copy the passage with my error, I highlight what I want with the mouse. I click on “edit” and then on “copy”. I can now go to `Kate` and paste the error in a file entitled, *LeFrMi* (i.e. “Learn From Mistakes”).

4. There are many other line commands, and often it is quicker to use these.

   `mkdir` (= make directory). You can create a sub-directory of the directory that you are in this way, instead of using `Konqueror`

   `rm` (= remove). To delete, instead of using `Konqueror`. If you do not want to be prompted write, “rm -f” where “f” is one of the “options” associated with the command “rm”.

   `rename`. Use this to change the names of a set of files all at once. [This is an example of a “batch” command which is an important use of terminals.] Suppose that by mistake you have created files named *project_01.pdf*, *project_02.pdf* ... and you want to replace the word “project” by the word “exercise”, you would type:

      `rename` project exercise *****.

   Here the first word is the wrong one, the second the correct one and the asterisk means “examine all files”.

   `more`. This used to quickly read a text file, a little at a time. Type “more *.bashrc*” to see all of the file *.bashrc* discussed below.

The shell file *.bashrc*

A Linux system contains many files which are usually not of direct interest to the user and so these files are “hidden” by putting a period before the name. If you type “cd” (to change to your home directory) and then “ls -a”, you will see the hidden files.

One the hidden files is [*.bashrc*]. Every user on a system has their own .bashrc file; the user “student” has one as does the user “root”. At start-up the system looks at [*.bashrc*] to see which special commands have been placed there. This is an example of a “shell file”. A “shell” is the collection of software that is used to communicate commands to the Linux operating system. Most Linux systems use “Bash” (= “Bourne Again SHell"; “Bourne” having been the name of a previous shell).

The file [*.bashrc*] is a good place to put your aliases, i.e. personalized short cuts. Theis what part of the pre-configured [.bashrc] file looks like. I can never remember the line command for leaving the terminal (it is “exit”), so I put three aliases: “quit”, “q” and “x”. Th e second group of aliases is for use when I forget to turn off the CAPS LOCK. Notice that comments are made by using #. Add to or change [*.bashrc*] to suit your taste. [You have to logout for the changes to take effect.]

```
alias cls= ‘‘clear’’ # clears console
alias quit= ‘‘exit’’ # exits console, su mod
alias q= ‘‘exit’’
alias x= ‘‘exit’’
alias copy= ‘‘cp’’

# captalized versions of above
alias MKDIR= ‘‘mkdir’’
alias CD= ‘‘cd’’
alias CLS= ‘‘clear’’
alias CP= ‘‘cp’’
alias Q= ‘‘exit’’
```

Personalized, system wide, shell files

Whereas *.bashrc* is meant for personalized commands,sometimes there are commands which many users would like to use or which an administrator want to keep users from changing. In `Student Linux` these are kept in the directory /usr/local/bin. Since everthing aside from /home/fsl student is unaccessible, you have to log in as “root” in order to change these.

Go to `Kate` $\Longrightarrow$ CTRL + O (or the Open icon) and keep on clicking on the up arrow until you arrive at /usr, then /usr/local and finally /usr/local/bin (Notice how many directories and sub-directories and sub-sub-directories there are.]. Open up the file *del*. This is a shell file that I created clean up things when using TEX or LaTEX. These two programs generate certain auxiliary files on the way to creating PDF filee. The first stage of TEXing creates a “dvi” (= “device independent”) file, the second stage transforms the dvi file into a PS (= “postscript”) file and the third stage transforms the ps file into a PDF (= “portable document format”). To simplify TEXing for `Student Linux` users, I have also created the shell file *dvi_letter* which combines the last two steps and then, using *del*, automatically erases all the remaining junk files. So when creating this PDF file I simply type in `Kate` on Desktop 2, then go to `Konsole` on desktop 3 and type: `tex` KONSOLE.tex and then `dvi_letter`. Then I switch to `Okular` on desktop, see my mistakes or changes that I want to make and then back to `Kate`.

Look at the other shell files to see what they do. Then modify as you wish (perhaps you want to list all the PDF files at end; just add `ls` ***.**pdf) or create your own.