10 -Die wichtigsten ersten Befehle

`cd` Absolut & Relativ,
`cd ~`
`ls`, `ls -la` (versteckte Dateien), (4)
`ls -lh` (h=human)
`ls -lR` (R=recursive)
`ls -ltr` (long, sorted by time, reverse: last modified file showed last)
`ls -lta` - Last Modified files are on top
`su`
`su -` (switch or substitute user) (5)
`cat` (concatenate) (6)
`whoami` - wer bin ich? :-)

(3) cd

Absolut: `cd /opt/kde`
`ls`
Relativ: `cd share`
`ls`
`cd ..`
`cd bin`
`ls`
`cd ../share`
`ls`
`cd doc`
`cd ../../bin`
Absolut:
`cd /opt/kde/share/applnk`

(4) ls

```
cd /opt/kde
ls
ls -l
cd ~
ls
ls -a
ls -al
ls /etc
ls /etc/XF*
ls -l /etc/XF*
```

mit YaST eine Gruppe einfügen
`ls -ltr /etc`

Commands for LPI-101

cut -d*x* -f*y* - Extracts columns from file: field(*y*) separator(*x*)
`eg. cut -d: -f1,6 /etc/passwd` (Extract field 1 and 6)

expand - Expands TABs to SPACES in text files.
eg. `expand /etc/init.d/at > ~/atnew`

fmt - Formatting of text files (useful for continuous text only)
eg. `fmt -w50 /usr/share/doc/packages/apache/ABOUT_APACHE`

head [-|+][n]*x* - Display first x lines of text file (default 10)
eg. `head -40 /etc/services`
(Display first 40 lines of file)

join - Joins lines of a data file on common fields
eg. `join -t: -11 -21 /etc/passwd /etc/shadow`

nl - Number the lines of a text file
`eg. nl -s" - " /etc/services`
Number the lines, adding " - " after line number

od -bih - Display file content in Octal(`-b`) Decimal(`-i`) Hexadecimal.(`-h`)
eg. `od -txCz /bin/ping`

paste - Pastes corresponding lines of 2 text files
eg. `paste /etc/passwd /etc/shadow`

pr - Convert text files for printing
eg. `pr /etc/services | less`

split -l*x* -b*y*[b|k|m] - Splits files into multi files containing (*x*)lines or (y) bytes,kilobytes or megabytes.
eg1. `split -l100 /etc/services serv`
creates `servaa servbb` etc. To get the original back
`cat serv?? > servicesnew`

eg2. `split -b1440k /bin/rpm rpms`
creates `rpmsaa rpmsab` etc. To get the original back
`cat rpms?? > rpmnew`

cat - Displays content of text file top to bottomand exits
eg. `cat -n /etc/hosts`
(show all lines of file with line numbers(`-n`)

tac - Displays content of text file bottom to top and exits
(reverse of `cat`)
eg. `tac /etc/passwd`
(List starts with the lasts users created in system)

`tail [-|+][n]x` - Display last *x* lines of text file (default 10)
eg1. `tail -30 /etc/services` (Display last 30 lines)
eg2. `tail +100 /etc/services`
(Bypass first 100 lines and display the rest till end of file)
eg3. `tail -fs5 --retry /var/log/httpd/error_log`
(read the last 10 lines of the file every 5 sec. and
keep retrying even if the file is not available)

`tr -d` - Translate or delete characters of file
eg1. `tr "a-z" "A-Z" < /etc/motd` (translates a-z to A-Z)
eg2. `tr -d "#" < /etc/services | less` (deletes all #)

`wc -c|-w|-l` - Counts number of lines(`-l`), words(`-w`) or chars(`-c`) of text file. Without options it shows all: lines, words and chars.
eg. `wc /etc/motd`

`xargs` - Reads text from pipe and provides it as parameter(s) for specified command.
eg. `find /etc -name *.conf | xargs cat > /root/confs`
Finds all `.conf` files in `/etc` and accumulates their contents all in one file called `/root/confs`.

`sed` - Stream file editor
eg. `sed 's/#/;-/g' /etc/services`
(see 45_Editing_Text-sed for more info.)

`more` - Forward only display of text file content
eg. `more -30 /etc/services`
(scrolls display next 30 lines when pressing space bar)
(press enter to scroll to the next line)

`less` - Scrollable display of text [file|pipe] content.
eg1. `less -X +G /etc/services`
(show file and leave the display as is(`-X`) when leaving less)
(and start at the end of the file (+G)
eg2. `less -phttps /etc/services`
(load file and go to first occurence of search pattern `https`)

`grep [-virns]` - Extract all lines of text where pattern is [not] found
eg1. `grep -ins "^f.p.*SSL$" /etc/services`
(Display all lines of file where pattern(ignoring case `-i`) is found with its line numbers(-n) and no error messages `-s`)
eg2. `ps -ax | grep httpd | grep -v grep`
(Display all instances of processes where httpd is found excluding(`-v`) the `grep httpd` command itself)

`sort -ky[n] -tx` - Sort text file by field(*y*) with separator(*x*)
eg. `ls -la | sort -k5n` *(sorted by size)*

`awk -Fx` - Programmable text formatter fields delimited (*x*)
`eg. awk -F: '{ print $1,"\t- ", $3 }' /etc/passwd`

`watch` : Repeats a command continuously `-n`x (min 1 sec. default 2 sec)

```
watch "pstree -p"
watch -n1 "ls -la"    # Repeat 'ls -la' every 1 sec
watch -n1 "route -n"
watch -n1 "netstat | grep tcp"
watch -n1 "who"
watch -n1 "showmount"
watch -n1 "finger"
```

`sort`

Sort-Merge Files

$sort [**-cmu**] [**-o***file*] [**-y***k*] [**-z***n*][**-dfiMnr**][**-btc**]
[*+pos1* **-***pos2*]] [*files*]

Description:

The **sort** command sorts lines from *files*.

Options:

-b	ignore leading tabs and spaces
-c	check that the input is in sorted order
-d	sort in dictionary order; only letters, digits, and white-space are significant in comparisons
-f	sort upper and lower case letters together
-i	ignore non-printable characters
-m	merge already sorted files
-M	sort as months. The first three non-blank characters are converted to upper case and compared. (implies **-b**)
-n	sort by numerical value; blanks, minus signs, and decimal points can also be given (implies **-b**)
-o*file*	send output to *file* (default standard output)
-r	reverse the sorting order
-t*c*	set the field separator to *c*
-u	display only one occurrence of duplicate lines
-y[*k*]	use *k* kilobytes of memory to sort (default max)
-z*n*	use *n* bytes of buffer for long lines
files	read standard input if *files* is **-** or no files given

+pos1 [**-***pos2*] sort from *pos1* to *pos2*.

If *pos2* is not specified, sort from *pos1* to the end of line. The format for pos1 and pos2 is: *m*[**.***n*] [**bdfinr**]

m	*m* fields from start of line (default **0**)
n	*n* characters from start of field (default **0**)
bdfinr	option applies to the specified key only

tr

Translate Characters

$tr [**-cds**] [*string1*] [*string2*]

Description:

The **tr** command copies standard input to output and translates characters from *string1* to characters in *string2*.

Options:

-c	translate characters not in *string1*
-d	delete characters in *string1* from input
-s	truncate repeated characters from *string2*

Strings:

[*a-z*]	specifies range of characters from *a* to *z*
[*c*n*]	specifies *n* repetitions of *c*. If the first digit in *n* is **0**, *n* interpreted as octal. (default is decimal)

---

uniq

Report Duplicate Lines

$uniq [**-udc** [*+n*] [*-n*]] [*file1* [*file2*]]

Description:

The **uniq** command removes duplicate adjacent lines from *file1* and places the output in *file2*.

Options:

-c	display a count of duplicate lines also
-d	display only duplicate lines once
-u	display only unique lines from the original file
-n	skip first *n* fields from start of line
+n	skip first *n* characters from the start of field

cut

Display File Fields

```
$cut -clist [ files ]
$cut -flist [ -dc ] [ -s ] [ files ]
```

Description:

The **cut** command displays fields from lines in the specified files according to selection options. The fields can be of fixed or variable length.

Options:

-c*list*	display characters from the positions in *list*
-d*c*	set the field delimiter to *c* (default tab)
-f*list*	display the fields specified in *list*
-s	suppress lines with no delimiter characters
files	read standard input if *files* are **-**, or no *files* are specified
list	comma separated list of integer field numbers; integers separated with a **-** indicate a range

---

echo

Display Arguments

```
$echo arguments
```

Description:

The **echo** command displays *arguments* on standard output. Special escape characters can be used to format arguments.

Escape Characters:

\b	backspace
\c	line without ending newline
\f	formfeed
\n	newline
\r	carriage return
\t	tab
\v	vertical tab
****	backslash
\0*x*	character whose octal value is *x*

grep

Search Files for Patterns

$grep [*options*] '*expression*' [*files*]

Description:

The **grep** command displays lines from *files* that match the given limited regular expression.

Options:

-b	precede each line with the block number
-c	display the number of matching lines
-i	ignore case of letters during comparisons
-l	display only filenames with matching lines once
-n	display the output with line numbers
-s	do not display error messages
-v	display non-matching lines only
files	read standard input if no *files* are specified

---

egrep

Search Files for Patterns

$egrep [*options*] '*expression*' [*files*]

Description:

The **egrep** command displays lines in *files* that contain the given full regular expression pattern.

Options:

-b	precede each line with the block number
-c	display the number of lines that match only
-e -*expression*	search for *expression* that begins with **-**
-f *file*	get expressions from *file*
-i	ignore case of letters during comparisons
-l	display file names with matching lines once
-n	display the output with line numbers
-v	display non-matching lines
files	read standard input if no *files* are specified

expr

Evaluate Expression Arguments

```
$expr arguments
```

Description:

The **expr** command evaluates *arguments* as an expression. Expression tokens must be separated with blanks, and special characters must be escaped. Integer arguments can be preceded by a minus sign to indicate a negative number.

Operators (listed in order of precedence):

exp1 | *exp2*	return *exp1* if neither null nor 0, else return *exp2*
exp1 \& *exp2*	return *exp1* if neither null nor 0, else return 0
exp1 \<, \<=, =, !=, \>=, \> *exp2*	return result of the integer or string comparison
exp1 +, -, *, /, % *exp2*	return result of the arithmetic operation
exp1 : *exp2*	return the result on the number of matched characters between *exp1* and *exp2*

---

paste

Merge Lines Between Files

```
$paste file1 file2 . . .
$paste -dlist file1 file2 . . .
$paste -s [ -dlist ] file1 file2 . . .
```

Description:

The **paste** command merges corresponding lines from *files*. Each *file* is treated as a column or columns of a table and displayed horizontally.

Options:

-d *list* replace tabs with characters from *list*. If this option is not specified, the newline characters for each file (except for the last file, or if **-s** is given, the last line) are replaced with tabs. The list can contain these special characters:

\n	newline
\t	tab
\0	empty string
\\	backslash

-s merge subsequent lines instead of one

files read standard input if *file1* or *file2* is **-**