ISCTE IUL
Instituto Universitário de Lisboa

Summary and Examples
Unix text tools

Consider the file `f1.txt`, with the following content:

```
Manuel Lisboa 0001-1002-12345234234-11 sim 110.0
Pedro Faro 0010-0302-00005234234-22 não 120.0
Maria Lisboa 0011-0333-00008989898-33 Sim 3333.5
Rui Lisboa 0100-0443-00004443442-32 sim 1223.0
Jorge Faro 0100-0443-00004444443-34 sim 232.5
Vanessa Porto 0110-0414-00004432442-31 sim 122.5
Joana Lisboa 0102-0414-00004444332-35 sim 456.0
Francisco Faro 0143-0424-00004423444-31 sim 12.5
```

Showing file content

`cat` .... shows the content of each file. `-n`: number the output lines; `-b`: number the non-blank lines
`less` .... shows files page to page, allowing forward and backward movement
`zcat, bzcat` .... decompress files and show their contents
`paste` .... show (merge) several files, side by side
Examples:
`cat f1.txt` .... shows the content of `f1.txt`
`cat f1.txt f2.txt > f3.txt` .... creates the file `f3.txt` with the content of `f1.txt` and `f2.txt`

wc

counts the number of (-w) words, (-l) lines, and (-c) characters.
`cat f1.txt | wc` .... shows the number of words, lines, and characters in f1.txt
`x=$( ls -l | wc -l )` .... the variable $x gets the number of files in the current directory
`words=$(echo "Batem leve, levemente" | wc -w)` .... variable $words gets the number of words in the sentence

head, tail

`head` and `tail` display the first/last *n* lines of a file. The output is 10 lines by default.
`cat f1.txt | head -5` .... shows the first 5 lines in f1.txt
`cat f1.txt | tail -5 > f2.txt` .... creates the file `f2.txt` containing the last 5 lines of `f1.txt`
`cat f1.txt | tail +5` .... shows all lines in the file, starting at line 5 until reaching the end
`cat f1.txt | head -3 | tail -2` .... shows lines 2 and 3 from the file f1.txt

cut

selects parts of a line. When applied to multiple lines, it extracts the columns.
`cat f1.txt | cut -d' ' -f3` .... shows only the bank account (NIB) for each person
`cat f1.txt | cut -d' ' -f3 | cut -d '-' -f2,3,4` .... shows columns 3,4 e 5 of each NIB
`cat f1.txt | cut -d' ' -f3 | cut -d '-' -f2-4` .... same affect as previous

sort, uniq

`sort` sort lines of text files. `uniq` filter out repeated lines in a file if the input
`cat f1.txt | cut -d ' ' -f2 | sort | uniq` .... shows the list of cities, sorted and without repetitions
`cat f1.txt | sort -k 2` .... sorts the file based on the city. `-k`: specifies the column to work with
`cat f1.txt | sort -n -k 5` .... sorts numerically, based on column 5. `-n`: stands for numerical sort
`uniq -c`: counts the number of times each line occurs

tr

Translates a set of characters into another set of characters.
`echo "nem uma folha bolia" | tr "a-z" "A-Z"` .... converts each lowercase character into uppercase
`cat f1.txt | tr " " "\t" | cut -f2` .... converts spaces into tabs, allowing to directly apply `cut`

comm

Compares two files line by line. It outputs 3 columns: a first with lines that only list1.txt contains, the second contains lines that only list2.txt contains, and the 3rd contains lines in common.

`comm -12 list1.txt list2.txt`	keeps only the 3rd column, containing lines in common
`comm -23 list1.txt list2.txt`	keeps only the 1st column, containing lines that only list1.txt contains

diff

`diff` compare files line by line, showing the differences between them

`diff f1.txt f2.txt`	compare the files `f1.txt` and `f2.txt`
`diff -i f1.txt f2.txt`	the same, ignoring the case
`diff -b f1.txt f2.txt`	the same, but ignores changes in the amount of white space
`diff -y f1.txt f2.txt`	output in two columns

Regular Expressions

symbol	**special meaning in Regular Expressions**
.	match any character including newline
[abc...]	match any characters in “abc...”, and may contain intervals
[^abc...]	match any characters except in “abc...”.
^	position at the beginning of a string
$	position at the end of a string
\c	match a literal character "c" even if "c" is meta-character by itself
\<	position at the beginning of a word
\>	position at the end of a word
$r*$	match zero or more regular expressions identified by r
$r+$	match one or more regular expressions identified by r
$r?$	match zero or one regular expressions identified by r
$r\{x\}$	the r expression is repeated x times. may include intervals (e.g. 3-5)
$r_1 \mid r_2$	match one of the regular expressions identified by r_1 or r_2.
$(r_1 \mid r_2)$	match one of the regular expressions identified by r_1 or r_2, treating them as one RE

grep

file pattern searcher. -E interpret pattern as an extended regular expression (ERE).

`cat f1.txt | grep "Faro"`	shows lines containing the sequence "Faro"
`cat f1.txt | grep "^[A-Z]....  "`	lines starting with uppercase words that contain a space after 4 letters
`cat f1.txt | grep -v "Lisboa"`	shows lines where "Lisboa" does not occur
`grep -n "Lisboa"`	each output line is preceeded by its relative line number in the file
`grep -i "lisboa"`	ignores case
`grep -w "Lisboa"`	the expression is searched for as a word

sed

stream editor. reads the input and produces a transformed output, according to the command received as argument.

`cat f1.txt | sed "s/sim/SIM/"`	replaces "sim" by "SIM", only once per line
`cat f1.txt | sed "s/[sS]im/SIM/g"`	replaces "sim" or "Sim" by "SIM"
`cat f1.txt | sed -E "s/  +/ /g"`	replaces multiple spaces by only one space
`cat f1.txt | sed -E "s/^ +//g; s/ +$//g"`	deletes spaces at the beginning and at the end of the string

awk

`awk` scans each input file for lines that match any of a set of specified patterns. Each pattern is associated with an action that will be performed when a line matches the pattern. An input line is normally made up of fields separated by white space, or by regular expression FS. The fields are denoted `$1`, `$2`, ..., while `$0` refers to the entire line.

`cat f1.txt | awk '{print $2, $1}'`	shows the first two columns of f1.txt in reverse order
`cat f1.txt | awk '/^J/ { print $2, $1 }'`	the same for lines starting with “J”
`cat f1.txt| awk -F "[- ]" '{print $1, $5}'`	separator defined as “-” or space. shows name and NIB’s 3rd column
`cat f1.txt | awk '{print "Campos:", NF, "Saldo:", $NF}'`	shows number of fields and last column
`cat f1.txt | awk '{print "Nome:", $1, "Resposta:", $(NF-1)}'`	shows name and 4th column
`cat f1.txt | awk 'BEGIN{i=0} {print "Linha:", ++i, "Contem:", $0}'`	shows numbered lines