UNIX - Commands

Input

Directories

Files

Two Files

File System

cd change to directory — `cd ~ # change to the home directory`

pwd prints name of current directory — `/home/auric.goldfinger/projects/secret/grandslam`

mkdir creates directories

-p creates directories with parent directories in one go

-v print a message for each creatad directory (to stderr)

rmdir deletes directories if they are empty

-p removes directories and their empty subdirectories in one go

-v output a diagnostic for every directory processed

find looks for files

find auxFile looks for files named "auxFile" in the current directory only — `find: auxFile: No such file or directory`

find -name auxFile looks for files named "auxFile" in the current directory and its subdirectories — `./subdir1/thatDir/auxFile` `./subdir1/auxFile` `./subdir1/auxFile`

find -maxdepth 2 -name auxFile looks for files named "auxFile" in the (sub)directories up to level 2 (current directory being level 1)

find /home -maxdepth 1 -name "james*" looks for files/directories whose names start with "james" in the directory '/home' (first level only) — `/home/james.bond` `/home/james.brown` `/home/james.cook` `/home/james.kirk`

find -empty returns only directories/regular files that are empty — `./subdir1/dirWithoutContent`

-type l returns only symbolic links — `./subdir1/auxFile`

find -amin +25 returns only files/directories accessed more than 25 minutes ago — `.` `./subdir1/auxFile`

find -atime -2 returns only files/directories accessed less than 48 hours ago — `.` `./subdir1` `./subdir1/thatDir` `./subdir1/thatDir/auxFile` `./subdir1/auxFile`

find -mmin 100 returns only files/directories with data modified exactly 100 minutes ago — `./subdir1`

find -perm 700 — `./subdir1/dirWithoutContent`

returns only files/directories where owner has all permissions, group and others have no permission

find -name *Content -delete

finds all files/directories whose names end with 'Content' and deletes them

find /bin/ch* -exec wc '{}' \;

finds all files with names starting with 'ch' in directory '/bin' and executes command 'wc' on them

```
134 1381 56984 /bin/chgrp
110 1183 54112 /bin/chmod
148 1489 59180 /bin/chown
```

find -name auxFile -type f -exec chmod go+r '{}' \;

finds all regular files named 'auxFile' and adds read permission for group and others on those files

tree displays directory contents as tree

```
films
|-- action
|   `-- firstBlood
|-- comedy
|   `-- supernaturalComedy
|-- ghostbusters
|-- ghostbusters2
`-- horror
    |-- slasher
    `-- zombie
6 directories, 6 files
```

-d only directories

```
films
|-- action
|-- comedy
|   `-- supernaturalComedy
`-- horror
    |-- slasher
    `-- zombie
6 directories
```

-a entries starting with '.'

```
.
|-- .bash_history
|-- .bash_profile
|-- .bashrc
|-- deleted
|   |-- dir1
|   `-- file1
4 directories, 18 files
```

-u prints username

```
films
|-- [1594337] action
|   `-- [nerd.gee] firstBlood
|-- [1594336] comedy
|   `-- [nerd.gee] supernaturalComedy
|-- [1594340] ghostbusters
|   `-- [1594344] slasher
6 directories, 6 files
```

-l follows symbolic links

--inodes prints inode number

```
films
|-- [1594337] action
|   `-- [1594346] firstBlood
|-- [1594336] comedy
|   `-- [1594340] supernaturalComedy
|-- [1594344] ghostbusters
|   `-- [1594335] slasher
6 directories, 6 files
```

ls list content of directories

-l detailed output

```
drwxr-xr-x 11 rod rod 12288 Jun 23 13:33  Downloads
drwxrwxr-x  3 rod rod  4096  May  8  23:23  workspace
-rw-rw-r--  1 rod rod   481   Feb 18 12:12  X.ini
```

-a or --a list all (also hidden files)

```
-rw-r--r--  1  rod rod 179   Nov 9 2011    .config
drwxr-xr-x 11 rod rod 12288 Jun 23 13:33  Downloads
-rw-r--r--  1  rod rod 179   Nov 9 2011    examples.desktop
drwxrwxr-x  3  rod rod 4096  May 8  23:23  workspace
-rw-rw-r--  1  rod rod 481   Feb 18 12:12  X.ini
```

-i prints inode number

```
1594379 bin
1594480 Deleted
1594382 file2
1594394 films
```

-R list subdirectories' contents recursively

```
.:
bin deleted file2 films
./deleted:dir1 dir2 dir3 file1 file2 file3 list.txt
./deleted/dir1:
./deleted/dir2: dir
./films:action comedy horror
./films/action:firstBlood
./films/horror/zombie:28DaysLater predator
```

cp copies files

-r copies directories and their contents/subdirectories recursively

-b makes backups of existing destination files first

mv moves or renames files

-i asks before an existing file is overwritten

cat concatenate file contents and print them on the screen

-E end each line with $

```
Remember, remember$
The fifth of November$
Gunpowder, treason, and plot.$
I see no reason why the gunpowder treason$
should ever be forgot.$
```

-s only the first one of repeated empty lines is printed

```
#!/bin/bash
clear
while [ true ]; do
  sleep 1
  echo -e "I am the shit!"
done
echo -e "I stopped being the shit."
```

-n numbers the lines printed

```
1 Remember, remember
2 The fifth of November
3 Gunpowder, treason, and plot.
4 I see no reason why the gunpowder treason
5 should ever be forgot.
```

echo display a line of text

-n do not output the trailing newline

-e enable interpreatation of backslash escapes

chmod sets/changes file permissions

chmod u+rw myFile adds read and write permision for file "myFile" to the owners permisions

chmod g-wx myFile removes write and execute permision for file "myFile" from the groups' permisions

chmod o-wx myFile removes write and execute permision for file "myFile" from the others' permisions

chmod a=r myFile

sets the permissions for myFile to read for all owner, group, and others (and nothing else)

-R sets permissions on files and directories recursively

chmod UGO myfile (User Group Other)
eg: chmod 754 myfile
where each octal number stands for rwx:
001 = 1 = --x = execute only
010 = 2 = -w- = write only
011 = 3 = -wx = write and execute
100 = 4 = r-- = read only
101 = 5 = r-x = read and execute
110 = 6 = rw- = read and write
111 = 7 = rwx = full

touch creates new files

```
touch file{1,2,3}
ls
file1 file2 file3
```

rm deletes files

-r deletes directories and their contents/subdirectories recursively

-i asks before every deletion

-v explain what is being done

ln creates file links

no option results in hard links

-s makes symbolic instead of hard links

-t puts the generated links into a specified directory

stat prints information on file status

```
File: `wordList'
Size: 6338 Blocks: 16 IO Block: 4096 regular file
Device: fd01h/64769d Inode: 1594466 Links: 1
Access: (0670/-rw-rwx---) Uid: ( 1064/auric.goldfinger) Gid: ( 100/users)
Access: 2012-07-13 11:09:46.000000000 +0100
Modify: 2012-06-22 13:18:01.000000000 +0100
Change: 2012-07-09 15:08:19.000000000 +0100
```

-t information is given in terse form

```
wordList 6338 16 81b8 2209 100 fd01 1594466
1 0 0 1342174186 1340367481 1341842410 4096
```

-f status of file system instead of file is reported

```
File: "wordList"
  ID: adc0b8501468b536 Namelen: 255 Type: ext2/ext3
Block size: 4096 Fundamental block size: 4096
Blocks: Total: 7246955 Free: 5622141 Available: 5254321
Inodes: Total: 1839600 Free: 1702146
```

cmp compares two files

file1 file2 differ: byte 94, line 7

-l prints byte numbers and divergent byte values for every difference to stdout

```
94 122 124
95 157 151
96 147 155
97 145 157
cmp: EOF on file2
```

-s only returns exit status and prints nothing

diff reports differences between two files

```
1c1
< AbcdEfghIjklmnOpqrstUvwxyz
---
> abcdefghijklmnopqrstuvwxyz
```

-i differences in case are ignored

-s reports if files do not differ — `Files alphabet and letters are identical`

df reports usage of disk space by file system

```
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
  10063176 7788844 1763932  82% /
/dev/mapper/VolGroup00-LogVol02
  20063832 6333316 28939236  24% /home
/dev/sda1 194442 18711 165602  11% /boot
tmpfs 517308 96 517284  1% /dev/shm
```

-h sizes are given in easily understandable units

```
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00 9.6G 7.5G 1.7G 82% /
/dev/mapper/VolGroup00-LogVol02 28G 6.0G 21G 23% /home
/dev/sda1 190M 18M 162M 11% /boot
tmpfs 500M 96K 500M 1% /dev/shm
```

-i information on inodes instead of sizes is given

```
Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/mapper/VolGroup00-LogVol00 638976 136235 502741 22% /
/dev/mapper/VolGroup00-LogVol02 1839600 133618 1704584 8% /home
/dev/sda1 50200 40 50160 1% /boot
tmpfs 128345 3 128342 1% /dev/shm
```

du estimates space usage by directories

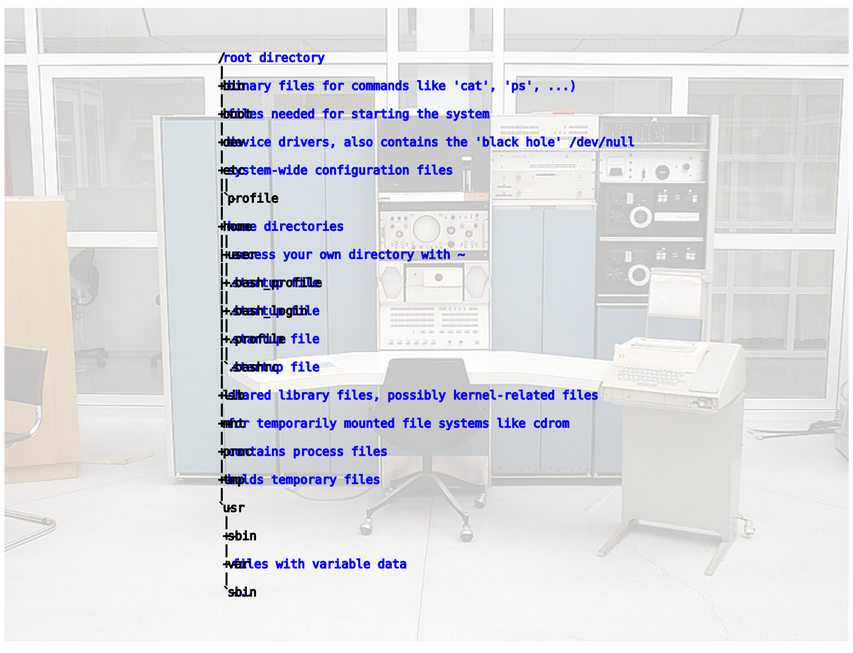
```
8 ./dir
8 ./films/horror/zombie
8 ./films/horror/slasher
24 ./films/horror
8 ./films/action
168 .
```

-s display total space usage only — `168.`

--time give also last modifaction time

```
8 2012-07-03 14:17 ./dir
8 2012-06-20 12:38 ./films/horror/zombie
8 2012-06-20 12:38 ./films/horror/slasher
24 2012-06-20 12:38 ./films/horror
168 2012-07-03 14:17 .
```

Files and Directories



Output/Input

Piping
The output of one command becomes the input of the other

`pwd | rev | cut -d "/" -f1 | rev`
displays the name of the current directory without the leading path

`ls | wc -w`
counts the files/directories contained in the current directory

`ps -e | wc -l`
counts the number of rows in the table that shows all processes

`ps -e | grep bash`
shows all processes that are shells

`ps -e | head`
displays the first 10 lines of the table that shows all processes

`history | grep "|"`
shows all commands in the history that involved piping

`ps -e | less`
displays the table that shows all processes in a scrollable view

`history | grep "|" | cut -d "|" -f2`
shows all commands behind a (first) pipe

`find peanuts 2> badReport`
the errors - if any - resulting from searching for peanuts are output to file 'badReport' instead of the screen at the end of file 'procList'

`find peanuts 1> result 2> badReport`
the output from searching for peanuts are written into file 'result' and the errors - if any - to file 'badReport' instead of the screen

`find peanuts 2>> failures 1>> found`
the output from searching for peanuts are appended to file 'found' and the errors - if any - to file 'failures' instead of being printed to the screen

File/Lines

Output Redirection

Output Redirection
of the channels
stdout = 1,
stderr = 2:
> Redirect stdout
2> Redirect stderr
2>&1 Redirect stderr to stdout
>> Append to stdout
2>&1 | Pipe stdout and stderr to another command

`echo "Obama" > president`
writes to file 'president' instead of the screen

`ps -ef > procList`
the table showing all processes is output to file 'procList'

`date >> procList`
current date and time is appended at the end of file 'procList'

Input Redirection

```
while read x; do
echo -e $(wc -w $x)
done < fileToBeRead
```

the content of file 'fileToBeRead' is used as input for 'read x', thus allowing to process 'fileToBeRead' line-by-line

$?, $#, ...

```
$var returns the value of variable
     var (but assignment is var=value)
$?   exit status of the last command/process
     (comparable to a return value), integer
     in { 0, 1, ..., 255 } (7); "true" is 0,
     "false" is 1 (in general, not 0);
     an exit status of 0 means 'success', anything
     different from 0 means ... well, something
     else; display exit status with echo $?
$#   number of parameters (strings separated by
     blanks) given to the command/script/method
$1, $2, ..., $9
     value of 1st, 2nd, ..., 9th parameter given
     to the command/script/method; works only for
     1, ..., 9, not for numbers with two or more
     digits (use shift operator in that case)
     CAUTION: for a script abc.sh containing a
     function xyz(), $1 can have different meanings
     inside abc.sh - outside of xyz() (and any
     other function), it means the 1st command
     line parameter given to abc.sh, e. g., "hello"
     in case of 'sh abc.sh hello world';
     inside of xyz(), it means the 1st parameter
     handed over to xyz(), e. g., "world" in case
     of 'xyz $2'. Analogously for $2, ..., $#, etc.
$$   process ID of the current shell
     (as displayed by 'ps')
$@   a list of all parameters given to the command/
     script/method, each one in its own " "
$*   a list of all parameters given to the command/
     script/method, all together in one pair of " "
$!   process ID of the last
     executed background command
```

Processing of Lines

rev displays characters in reverse order line-by-line

rev EinsteinQuote — `hguone llew ti dnatsrednu t'nod uoy ylpmis ti nialpxe t'nac uoy fI`

wc counts lines, words, characters

`80 171 935 hangman`

-l number of lines — `80 hangman`

-w number of words — `171 hangman`

-c number of characters — `935 hangman`

cut selects and displays sections from lines in a file

cut -c1-7 PrimeMinisters selects the 1st to 7th character of every line in file"PrimeMinisters"

```
Tony Bl
Gordon
David C
```

cut -f1 -d"-" PrimeMinisters selects 1st field of every line in file "PrimeMinisters", interpreting '-' as field delimiter

```
Tony Blair|1997
Gordon Brown|2007
David Cameron|2010
```

cut -f1,3 -d "|" --output-delimiter="; " PrimeMinisters selects 1st and 3rd field of every line in file "PrimeMinisters", interpreting '|' as field delimiter and replacing it by '; ' for the output

```
Tony Blair; Labour
Gordon Brown; Labour
David Cameron; Tories
```

grep searches in files for lines matching a specified pattern, and prints the results

grep Tories PrimeMinisters searches for lines containing 'Tories' in file "PrimeMinisters"

```
Maggie Thatcher|1979-1990|Tories|1979, 1983, 1987
John Major|1990-1997|Tories|1992 elections
David Cameron|2010-20??|Tories|2010 elections
```

grep -c Labour PrimeMinisters searches for 'Labour' in file "PrimeMinisters", prints only a count of matching lines found — `2`

grep -f FourLetterNames PrimeMinisters searches in file "PrimeMinisters" for patterns read from file "FourLetterNames"

```
John Major|1990-1997|Tories|1992 elections
Tony Blair|1997-2007|Labour|1997, 2001, 2005 elections
```

grep -i G PrimeMinisters searches case-insensitive for 'G' in file "PrimeMinisters"

```
Maggie Thatcher|1979-1990|Tories|1979, 1983, 1987
Gordon Brown|2007-2010|Labour|elections
```

grep -n Tony PrimeMinisters searches for 'Tony' in file "PrimeMinisters", prints also the numbers of the lines found — `3:Tony Blair|1997-2007|Labour|1997, 2001, 2005 elections`

grep -v Tories PrimeMinisters selects and prints only those lines of file "PrimeMinisters" not matching the pattern 'Tories'

```
Tony Blair|1997-2007|Labour|1997, 2001, 2005 elections
Gordon Brown|2007-2010|Labour|elections
```

tee writes input to stdout and a file at the same time

tee tFile writes the input of the user into file "tFile" as well as to the terminal

```
Hello, this is me writing some text.
Hello, this is me writing some text.
Why did that repeat?
Why did that repeat?
^C
```

tee -a tFile appends the input to file "tFile" instead of overwritting it

```
And more text. Will this be added to the content of "tFile" instead of replacing it?
And more text. Will this be added to the content of "tFile" instead of replacing it?
Yes it does!
^C
```

head prints the first 10 lines to stdout

```
line 1
:
line 10
```

-n prints the first n lines

```
line 1
:
line n
```

tail prints the last 10 lines to stdout

```
line 19
:
line 29
```

-n prints the last n lines

```
line 19
:
line 19+n
```

more loads a whole file and prints its content to the screen one page at a time

less prints a file's content to the screen in a scrollable view, loading only as much of the file as needed

sort prints sorted file contents to stdout

sort Beatles sorts lines in file "Beatles" and prints them out

```
Harrison, George, 1943, 2001
Lennon, John, 1940, 1980
McCartney, Paul, 1942, 2077
Starr, Ringo, 1940, 2077
```

sort -r Beatles sorts lines in file "Beatles" in reverse order and prints them out

```
Starr, Ringo, 1940, 2077
McCartney, Paul, 1942, 2077
Lennon, John, 1940, 1980
Harrison, George, 1943, 2001
```

sort -c Beatles does not sort but check if lines in file "Beatles" are sorted — `sort: Beatles:3: disorder: Harrison, George, 1943, 2001`

sort -n Commodities sorts lines of file "Commodities" numerically

```
27.5$/oz.       Silver
106.5$/bbl.     Oil
1600$/oz.       Gold
```

sort -k 2 Commodities sorts lines of file "Commodities", the sort key is understood to start 2nd field

```
1600$/oz.       Gold
106.5$/bbl.     Oil
27.5$/oz.       Silver
```

uniq prints lines to stdout omitting those who are copies of their predecessor

uniq USPresidentialElectionWinners prints lines of file "USPresidentialElectionWinners" to stdout omitting repetitions

```
George Bush
Bill Clinton
George W. Bush
Barack Obama
```

uniq -c USPresidentialElectionWinners prints non-double lines of file "USPresidentialElectionWinners" to stdout, together with the number of occurrences

```
1 George Bush
2 Bill Clinton
2 George W. Bush
1 Barack Obama
```

uniq -u USPresidentialElectionWinners prints only non-repeated lines of file "USPresidentialElectionWinners" — `George Bush` `Barack Obama`

uniq -d USPresidentialElectionWinners prints only repeated lines of file "USPresidentialElectionWinners"

```
Ronald Reagan
Bill Clinton
George W. Bush
```

tr replaces specified characters by given substitutes

echo "Mind the gap." | tr Mg Fm? replaces 'M' by 'F' and 'g' by 'm' in "Mind the gap." — `Find the map.`

echo "West End" | tr -d sn specified characters 's', 'n' are not replaced but deleted in "West End" — `Wet Ed`

echo "Summer 1999" | tr -s m9 p% any sequence of 'm's, '9's in "Summer 1999" is replaced by one 'p' or '%', respectively — `Super 1%`

Processes and OS

ps displays a table of current processes

```
PID TTY TIME CMD
 7319 pts/22 00:00:00 bash
23321 pts/22 00:00:00 ps
```

-e list all processes

```
PID TTY TIME CMD
  1 ? 00:00:07 init
  2 ? 00:00:00 kthreadd
  .
  .
25139 ? 00:00:00 script
25140 pts/61 00:00:00 bash
```

-f gives a full format listing

```
UID PID PPID C STIME TTY TIME CMD
1234 7319 7318 0 09:40 pts/22 00:00:00 -bash
1234 25852 7319 1 11:35 pts/22 00:00:00 ps -f
```

top displays dynamic real-time view of current Linux tasks

-d 1.5 sets the delay between screen updates to 1.5 seconds

& puts the command before in the background — `find / -name "abc" &`

jobs lists active jobs

-l gives also process IDs — `[1]+ 8020 runs find / -name "abc" &`

-r lists only running jobs — `[1]+ runs find / -name "abc" &`

-s lists only stopped jobs

bg After suspending a job with [CTRL-Z], resumes execution of the suspended process, leaving it in the background

fg resumes execution of a suspended process by bringing it to the foreground and thus redirecting its standard input and output streams to the user's terminal — `fg %1 # sends job #1 to the foreground`

fork SYSTEM CALL: generates a child process as 'copy' of the current one, its parent

exec SYSTEM CALL: a process starts its work, the execution of its associated command

exit SYSTEM CALL: a process terminates after doing its work, causing its parent to 'wake up', and returning an integer as exit status

wait SYSTEM CALL: the calling process 'goes to sleep' in order to allow its child(ren) to do their work

kill terminates processes specified by process Ids

-l lists all signals that 'kill' can send

-p simply prints the process IDs to stdout

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
```

killall terminates all processes with matching names

-u terminates all processes of the specified user

-w waits until all relevant processes are dead

-l lists all signals that can be sent

```
HUP INT QUIT ILL TRAP ABRT IOT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM TERM
STKFLT CHLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF WINCH IO PWR SYS
UNUSED
```

sudo execute the following command as a superuser — `sudo rm -r *` `[sudo] password for till:`

su become the superuser for a session

```
su                password:
rm -r *
```

finger displays information about users currently logged in

```
Login Name Tty Idle Login Time Office Office Phone
theboy.whoklived Harry Potter pts/10 Jul 18 11:17 (abc0112358.fdmgroup.local)
youknow.who Tom Riddle pts/2 Jul 18 11:03 (xyz235711.fdmgroup.local)
albus.dumbledore Albus Dumbledore pts/0 8 Jul 18 09:56 (headmaster.hogwarts.wiz)
```

-l gives more detailed information about users

```
Login: ron.wea Name: Ronald+ Weasley
Directory: /home/ron.wea Shell: /bin/bash
On since Wed Jul 18 09:56 (BST) on pts/0 from 11.5.9.139
  14 minutes 21 seconds idle
No mail.
No Plan.
```

who shows users logged on

```
michael.jackson pts/0 2009-06-25 09:56 (11.2.7.151)
james.roberts pts/4 2012-07-18 10:54 (abc0815.fdmgroup.local)
jack.rackham pts/10 2012-07-18 11:17 (lapfrk2012.fdmgroup.local)
```

-q displays usernames and a count

```
michael.jackson
james.roberts
jack.rackham  # users=3
```

-b displays time of last system boot — `system boot 2012-05-27 11:47`

which shows the path of executable files associated to commands — `/usr/bin/finger`

-a shows all paths, not only the first one

```
/bin/cut
/usr/bin/cut
```

env print the environment variables to the screen

```
HOSTNAME=unix
SHELL=/bin/bash
USER=you.there
```

man manual for command

info info about command

date print date

cal print calendar

free print free memory

uptime print uptime

clear clears the screen

exit ends the current shell

exit 66 ends the current shell with an exit status of 66 (can be displayed by excecuting 'echo $?' afterwards)

sleep delay for a specified amount of time

history all the commands entered so far

Scripting

```
#!/bin/bash                          # this is called she-bang
for x in 0 1 2 3 4 5 6 7 8 9; do #for every item x in the list do the following:
   if [ $x -eq 0 ]; then                #  if x = 0 ...
                  echo -e "$x reads zero "
   elif [[ $x -lt 3 || $x -eq 6 ]]; then #  ... else, if x < 3 or x = 6 ...
                  echo -ne "$x = three "
                  sleep 2
                  echo -ne "letters in one word\n"
   elif [[ $x -ne 3 && $x -ne 8 && $x -ne 7 ]]; then # ... else, if x is not 3, 8 or 7 ...
                  echo -e "$x is a four-letter word"
   else        # ... else (any case left)
                  echo -e "$x is five characters long"
   fi                                   #  end of if branches
done                                    #  done! loop is finished.
```

```
while read line #  as long as 'read line' works (i. e., returns 0),
   do                                   #  do the following:
      first=$(echo -n $line) | cut -c1
      case $first in                    #  in the case that first is ...
        "[A-Z]") echo -ne "$line\t starts with a capital letter\n";;     #  ... a capital letter
        "[a-z]") echo -ne "$line\t starts with a lower case letter\n";;  #  ... a lower-case letter
        "[0-9].*") echo -ne "$line\t starts with a number\n";;           #  ... a numerical
        *) echo -ne "$line\t starts with a non-alphanumeric character\n";; #  ... anything else
      esac                              #  end of case branches
 done < fileWithLines # input for 'read line' (cf. above) is taken from file 'fileWithLines'
```

```
    x=1
until [ $x -ge 999999 ]; do                 #  as long as NOT x >= 999999, do the following:
                  echo -e "$x is not enough, bid more!"
                  x=$[$x*10]
  done                                      #  done! loop is finished.
```

```
Numerical comparison
        if [ $x -eq 1 ]; then  // if x = 1     if [ $x -ne 2 ]; then # if not x = 2
        if [ $x -ge 3 ]; then  // if x >= 3    if [ $x -gt 4 ]; then # if x > 4
        if [ $x -le 5 ]; then  // if x <= 5    if [ $x -lt 6 ]; then # if x < 6
  String Comparison
        if [ $w = "London"     ]; then # if w is exactly the string "London"
        if [ $w != "Brighton" ]; then # if w is not exactly the string "Brighton"
```

Generated with the Haskell diagrams library