basics.sh

```
#!/bin/bash

#this file is not intended to be executed
#the file has a form of a bash script just for the syntax highlighting
exit 0 #in case someone tries to execute the script just exits doing
nothing

#########
# The syntax
#########
command [arg1] [arg2] [arg3] ... [argN]<ENTER>
#example
cp source.txt destination.txt

############
# The options
############
# -the option is an argument too
# -modifies commands behavior
# -can have a long  "--abc-def" or a short "-a" variant
# -the option can has its own arguments, usually separated by white
spaces
ls
ls -r
ls -x
ls -xr
ls -rx
df -h
df --human-readable
gcc -o prog prog.c
###################################
#how to deal with arguments starting with "-" sign
###################################
# if someone creates a filename starting with "-"
# !!!NEVER CREATE SUCH FILES!!!
#...but in case someone still creates one "--" is the explicit end of
options

ls -- -l #can deal with the file named: "-l"
#or
ls ./-l
ls -l -- -l

################
# Dealing with input
################
# when keys are pressed by user the terminal examines the keys one by
one and determines how to react
# you can use, HOME, END, or Backspace, these actually do something
```

```
else than  just printing the character
# try also:
# ctrl+h
# ctrl+u
# ctrl+w
# characters are stored into the buffer if no special reaction is
required
# the buffer is displayed on the screen, except the nonprintable
characters (white space, tab)
# press ENTER to pass the buffer content to the shell

###############
# Execution order
###############
#the command does not have to be on the first position:
>bb <aa cat    #runs the "cat" command using "aa" file as input source
and "bb" as output
        #ofc there is no reason for this rather unclear form
cat <aa >bb    #works just fine as well

########################
# Display as a file
########################
# standard output is a special file "device file"
# you can threat it as regular file in some cases
# who command tells you what is your actual tty
# you can than try to write something to that file
# may not be tty if you are using terminal emulator instead of true TTY
jano@panoramix:~$ who
jano     tty1         2015-11-08 10:18
jano     tty7         2015-11-05 08:46 (:0)
jano@panoramix:~$ sudo su
[sudo] heslo pro jano:
root@panoramix:/home/jano# echo Hello > /dev/tty1
Hello
root@panoramix:/home/jano#


#########################
#Standard Input and Output
#########################
# explore the cat command with no arguments

jano@panoramix:~$ cat
Hello[ENTER]
Hello
Hello again[Ctrl+d]Hello Again[Ctrl+d]jano@panoramix:~$

#Ctrl+d is a special character "EOF" = End Of FIle
```

```
##############
# Redirection
##############
#stdin is normally taken from keyboard and the stdout is printed on the
screen
#this behavior can be modified by redirection

############################
# output redirection ">"
############################
command [arg1] [arg2] ... [argN] > filename

jano@panoramix:~$ cat > outfile.txt
This is the text I write[Enter]
here after the enter key[Enter]
and more[Enter]
now preess Ctrl+d to end[Eter]
end[Ctrl+d]
jano@panoramix:~$

# the cat command can be used as an "emergency" text editor

#cat can be used to merge files as well
jano@panoramix:~$ cat > one.txt
first
jano@panoramix:~$ cat > two.txt
second
jano@panoramix:~$ cat > three.txt
third
jano@panoramix:~$ cat one.txt two.txt three.txt > alltogether.txt
jano@panoramix:~$ cat alltogether.txt
first
second
third
jano@panoramix:~$
#to split the file you can use "split" command (man split)

# many programs behave the same way as cat
# lpr, sort, grep,....
# if there is no explicit input/output specified in the arguments
# the stdin/stdout is used by default

###################
# Beware of overwriting!!
###################
# a new file is crated by shell if there is no one, it is ERASED in
case it exists already!!!
# ofc the permissions apply
# this can be modified by setting the "noclobber"
# or by using a pipe sign "|"
```

```
jano@jano-MS-7793:~$ set -o noclobber
jano@jano-MS-7793:~$ echo Hello > tmp.txt
jano@jano-MS-7793:~$ echo Hello > tmp.txt
bash: tmp.txt: cannot overwrite existing file
jano@jano-MS-7793:~$ set +o noclobber
jano@jano-MS-7793:~$ echo Hello > tmp.txt
jano@jano-MS-7793:~$

###################
#Warning!!!
###################
jano@panoramix:~$ cat one.txt two.txt three.txt >| one.txt
jano@panoramix:~$ cat one.txt
second
third
jano@panoramix:~$
#the shell erases the file one and writes the content of two and three
there,.... is that what we were looking for?

####################
# appending
####################

jano@panoramix:~$ whoami > WhoAndWhen.txt
jano@panoramix:~$ date >> WhoAndWhen.txt
jano@panoramix:~$ cat WhoAndWhen.txt
jano
St lis  8 12:56:49 CET 2015
jano@panoramix:~$

#################################
# Writing to nowhere and reading nothing
#################################
# /dev/null is a special device file
# anything written there simply gets lost
jano@panoramix:~$ echo "This text is not stored anywhere" > /dev/null
# returns EOF in case of being read
jano@panoramix:~$ cat /dev/null
jano@panoramix:~$

#######
# Pipes
#######
# the pipe sign "|" can chain commands so the output of previous one is
taken as input to the next one
jano@panoramix:~$ cat letters.txt
ALL CAPITALS
jano@panoramix:~$ cat letters.txt | tr ABCDEFGHIJKLMNOPRSTUV
abcdefghijklmnoprstuv
all capitals
jano@panoramix:~$ cat letters.txt | tr ABCDEFGHIJKLMNOPRSTUV
```

```
abcdefghijklmnoprstuv > small.txt
jano@panoramix:~$ cat small.txt
all capitals
jano@panoramix:~$
#tr = translates the text substituting letter by letter according to
the template

# more pipes in a row are called a "filter"
jano@panoramix:~$ cat letters.txt | tr ABCDEFGHIJKLMNOPRSTUV
abcdefghijklmnoprstuv | wc > stat_small.txt
jano@panoramix:~$ cat stat_small.txt
      1       2      14
jano@panoramix:~$

###########
# tee
##########
# tee can deflect the content to a file in the middle of a filter
without interrupting it
jano@jano-MS-7793:~/tmp/bashexamples$ dmesg | grep mouse
[    1.538598] mousedev: PS/2 mouse device common for all mice
jano@jano-MS-7793:~/tmp/bashexamples$ dmesg | grep mouse | tee
mouselog.txt | wc
      1      10      63
jano@jano-MS-7793:~/tmp/bashexamples$ cat mouselog.txt
[    1.538598] mousedev: PS/2 mouse device common for all mice
jano@jano-MS-7793:~/tmp/bashexamples$

################
# Process management
################
# a program/process occupies the keyboard and the display if not
specified otherwise
# we do not want/need this sometimes
# process can be sent to background by adding &
#
jano@panoramix:~$ ping google.com &
jano@panoramix:~$ PING google.com (216.58.201.78) 56(84) bytes of data.
64 bytes from prg03s01-in-f78.1e100.net (216.58.201.78): icmp_seq=1
ttl=56 time=1.81 ms
64 bytes from prg03s01-in-f78.1e100.net (216.58.201.78): icmp_seq=2
ttl=56 time=0.607 ms
^C #ctrl+c does not kill the ping since it does not listen to kb
anymore
jano@panoramix:~$ 64 bytes from prg03s01-in-f78.1e100.net
(216.58.201.78): icmp_seq=8 ttl=56 time=0.589 ms
64 bytes from prg03s01-in-f78.1e100.net (216.58.201.78): icmp_seq=9
ttl=56 time=0.587 ms
64 bytes from prg03s01-in-f78.1e100.net (216.58.201.78): icmp_seq=10
ttl=56 time=0.615 ms
...
```

```
fg # ping listens to the kb once again (foreground)
64 bytes from prg03s01-in-f78.1e100.net (216.58.201.78): icmp_seq=11
ttl=56 time=0.591 ms

ping google.com
64 bytes from prg03s01-in-f78.1e100.net (216.58.201.78): icmp_seq=12
ttl=56 time=2.55 ms
64 bytes from prg03s01-in-f78.1e100.net (216.58.201.78): icmp_seq=13
ttl=56 time=0.600 ms
^C # ctrl+c now works
--- google.com ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12004ms
rtt min/avg/max/mdev = 0.587/1.015/2.552/0.597 ms
jano@panoramix:~$

#Ctrl+z stops the process
#bg sends it to the background (let it run again)
#fg brings it back to foreground
jano@panoramix:~$ ping google.com
PING google.com (216.58.201.78) 56(84) bytes of data.
64 bytes from prg03s01-in-f14.1e100.net (216.58.201.78): icmp_seq=1
ttl=56 time=1.87 ms
64 bytes from prg03s01-in-f14.1e100.net (216.58.201.78): icmp_seq=2
ttl=56 time=0.870 ms
64 bytes from prg03s01-in-f14.1e100.net (216.58.201.78): icmp_seq=3
ttl=56 time=1.85 ms
^Z
[1]+  Stopped                 ping google.com
jano@panoramix:~$ bg
[1]+ ping google.com &
jano@panoramix:~$ 64 bytes from prg03s01-in-f14.1e100.net
(216.58.201.78): icmp_seq=4 ttl=56 time=2.24 ms
64 bytes from prg03s01-in-f14.1e100.net (216.58.201.78): icmp_seq=5
ttl=56 time=0.908 ms
64 bytes from prg03s01-in-f14.1e100.net (216.58.201.78): icmp_seq=6
ttl=56 time=1.76 ms
fg
ping google.com
64 bytes from prg03s01-in-f14.1e100.net (216.58.201.78): icmp_seq=15
ttl=56 time=2.07 ms
64 bytes from prg03s01-in-f14.1e100.net (216.58.201.78): icmp_seq=16
ttl=56 time=0.591 ms
^C
--- google.com ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 19144ms
rtt min/avg/max/mdev = 0.580/1.339/2.460/0.680 ms
jano@panoramix:~$

#The process running in the background does not occupy our keyboard but
still sends its output to the screen
#the output can be redirected to null if we are not interested in it
```

```
jano@panoramix:~$ ping google.com > /dev/null &
[1] 11089
#ping does its job in the background and occupies neither screen or
keyboard
#actually it still kind of occupies the screen (stderr is not
redirected yet)
jano@panoramix:~$ killall ping
[1]+  Ukončen (SIGTERM)      ping google.com > /dev/null
jano@panoramix:~$
#or
jano@panoramix:~$ ping google.com > /dev/null &
[1] 11625
jano@panoramix:~$ jobs
[1]+  Running                 ping google.com > /dev/null &
jano@panoramix:~$ fg 1
ping google.com > /dev/null
^Cjano@panoramix:~$

#or by ps and kill-9
jano@panoramix:~$ ping google.com > /dev/null &
[1] 12717
jano@panoramix:~$ ps -A | grep ping
12717 pts/5    00:00:00 ping
jano@panoramix:~$ kill -9 12717
jano@panoramix:~$

##########################
#stderr - standard error#
##########################

student@user-ThinkCentre-E73:~$ ping totalabsolutenonsense.com >
/dev/null &
[6] 7029 #output is redirecdted to null but program can still
complain(standard error output) to the screen
student@user-ThinkCentre-E73:~$ ping: totalabsolutenonsense.com: Name
or service not known

student@user-ThinkCentre-E73:~$ ping totalabsolutenonsense.com >
/dev/null 2> /dev/null &
[6] 7085 #now we do not see the complain
student@user-ThinkCentre-E73:~$

student@user-ThinkCentre-E73:~$ ping totalabsolutenonsense.com >
/dev/null 2>&1 &
[1] 7334 #use the same destination for both (useful for an actual to
file redirection too ofc)
student@user-ThinkCentre-E73:~$

student@user-ThinkCentre-E73:~$ ping totalabsolutenonsense.com &>
/dev/null &
[1] 7338 #use the same redirect both but shoter
```

```
student@user-ThinkCentre-E73:~$


#######
# nohup
#######
#nohup lets a process run in the case of TERM signal

jano@panoramix:~$ echo $$ #first bash - the parent
17189
jano@panoramix:~$ bash # second bash - parent of ping
jano@panoramix:~$ echo $$
25013
jano@panoramix:~$ nohup ping google.com > /dev/null 2>/dev/null & #ping
to background
[1] 25050
jano@panoramix:~$ exit # terminates the parent, his child would die too
normally
exit
jano@panoramix:~$ ps -A | grep ping  #but not this time (nohup)
25050 pts/5    00:00:00 ping
jano@panoramix:~$ echo $$
17189
jano@panoramix:~$ kill -9 `ps -A | grep ping | awk '{print $1}'`
#(`...` a sub-shell more about that later)
jano@panoramix:~$ ps -A | grep ping
jano@panoramix:~$

##############################################
#want to nohup an already running process?#
##############################################
#go study "disown"

student@user-ThinkCentre-E73:~$ bash
student@user-ThinkCentre-E73:~$ ping root.cz 2>&1 > /dev/null &
[1] 7792
student@user-ThinkCentre-E73:~$ jobs
[1]+  Running                 ping root.cz 2>&1 > /dev/null &
student@user-ThinkCentre-E73:~$ disown -h %1
student@user-ThinkCentre-E73:~$ exit
exit
student@user-ThinkCentre-E73:~$ ps -A | grep ping
   7792 pts/2    00:00:00 ping #ping still running after parent bash is
dead
student@user-ThinkCentre-E73:~$

########
#screen#
########
#for more complex detachable terminal
```

```
user@machine:~$ apt install screen

#Basic usage:
# just run "screen", get the shell, work there as usual
# once done you can "detach the screen" <ctrl>+<a> & <d>
# now the screen is "detached" and you should be back in your original
shell (the one from which you have run the screen previously)
# your work on the screen-screen is not lost, you can restore it
running "screen -r"
# you can give a name to your screen session "screen -S
my_session_name"
# then restore the exact one (yes, you can have multiple screens)
"screen -r my_session_name"

#Reading history:
# to read the history of a terminal in screen, attach it and press
<ctrl>+<a> & <[>
# this enters to so-called "copy ode" now you can navigate via arrows
or PgUp/Down
# get back to standard mode by pressing <[> again


########
# tmux #
########
# Simmilar to screen and even more powerful is tmux
# go read about it:
# https://www.redhat.com/sysadmin/introduction-tmux-linux

##########################################################################
###
#more process killing magic examples using pgrep, pkill, killall, xargs
:)
##########################################################################
###
jano@panoramix:~$ ps -A | grep bash | awk '{print $1}'
5854
5930
13475
jano@panoramix:~$ pgrep bash
5854
5930
13475
jano@panoramix:~$

#kill that ping

jano@panoramix:~$ nohup ping google.com > /dev/null 2>/dev/null &
[3] 17768
jano@panoramix:~$ nohup ping google.com > /dev/null 2>/dev/null &
[4] 17769
```

```
jano@panoramix:~$ pgrep ping
17730
17768
17769
jano@panoramix:~$ kill $(pgrep ping)
jano@panoramix:~$
#command $(command $(command [arg1] [arg2]...)) more  sophisticated
sub-shell variant
#the same as `command`


# or
jano@panoramix:~$ ps -A | grep ping | awk '{print $1}' | xargs kill -9
# or
jano@panoramix:~$ pkill ping
# or
pgrep ping | xargs kill
# or
killall ping
# or even use a simple script (more commands in a loop)
jano@panoramix:~$ for proc in $(pgrep <process command>); do kill
$proc; done

#######################################
#Wildcards
#######################################
# you can use Wildcards in filenames
# these are called ambiguous file links
# so called "globbing" is applied to those


# ?
# exactly one char
jano@jano-MS-7793:~/tmp/bashexamples$ ls
file1  file2  file3  file4  file45  file46  file47
jano@jano-MS-7793:~/tmp/bashexamples$ ls file?
file1  file2  file3  file4
jano@jano-MS-7793:~/tmp/bashexamples$
#shell generates the list of all files matching the pattern and uses it
as a ls parameters

#use echo if unsure what will be generated
jano@jano-MS-7793:~/tmp/bashexamples$ echo file?
file1 file2 file3 file4
jano@jano-MS-7793:~/tmp/bashexamples$

#if there is no match, either the pattern itself is passed to the
command or an empty string depends on "nullglob" setting
jano@jano-MS-7793:~/tmp/bashexamples$ shopt -u nullglob
jano@jano-MS-7793:~/tmp/bashexamples$ echo file???
file???
```

```
jano@jano-MS-7793:~/tmp/bashexamples$ shopt -s nullglob
jano@jano-MS-7793:~/tmp/bashexamples$ echo file???

jano@jano-MS-7793:~/tmp/bashexamples$

# *
# any number of characters
jano@jano-MS-7793:~/tmp/bashexamples$ echo file*
file1 file2 file3 file4 file45 file46 file47
jano@jano-MS-7793:~/tmp/bashexamples$

# [...]
# the letter in brackets represent itself
# ranges can be used too
jano@jano-MS-7793:~/tmp/bashexamples$ echo file[1-9]
file1 file2 file3 file4
jano@jano-MS-7793:~/tmp/bashexamples$ echo file[1-9][1-9]
file45 file46 file47
jano@jano-MS-7793:~/tmp/bashexamples$ echo file[123456789][1-9]
file45 file46 file47
jano@jano-MS-7793:~/tmp/bashexamples$
#the match can be inverted by ^ or !
jano@jano-MS-7793:~/tmp/bashexamples$ echo file[!2]
file1 file3 file4
jano@jano-MS-7793:~/tmp/bashexamples$



#################
# Scripting
#################
# bash script is the ordinary(text) file containing a bunch of shell
commands
# there has to be a link to interpret at the beginning

#!/bin/bash (in case of bash script)

# it has to be executable
jano@desktop:~/ chmod +x thescript.sh
jano@desktop:~/
#can have .sh suffix for lucidity

# plain script can be made just by writing more commands in an order
just like if they were written in a command line
# no need to separate commands by ; if one command per line is used

#EXERCISE
#List all the character devices in /dev
#List the files newer than one week
#Create a file in your home dir that your classmate can view but
prevents him from viewing the rest of your home dir.
```

```
#Create a file in your home directory named SystemStats.txt (may be on
k332 or your local machine). Fill the file with the following info:
  #the total amount of RAM memory
  #disk free space on /home filesystem
  #CPU model name
  #explicit information about the used architecture being 64bit yes/no
  #explicit information about hyperthreading being available yes/no
#Create a bash script which when executed creates the upper
SystemStats.txt file in the users home directory.
```