

Linux cheatsheet

uname
who
whois
pwd
date
cal

ls -ld /usr/???
[gC]*
{1..10}

alias lh="ls -lh"

man passwd / man -k SHA1
info passwd
whereis bzip2
which bzip2
type echo / type ls

ls / ls -la / ls -lR / ls -ld
ln / ln -s
du -h
head / head -n-1 / head -n5
tail
cat
less
wc -l
nl
cut -d : -f 2

tr -s "<,>" "[,]"
tr -d "@"
tr -s "[:lower:]" "[:upper:]"
sort
uniq / uniq -D
touch

grep / grep -vE '^ii.*' / grep -i
grep -E "1[0-9]{2}-[0-9]{3}" file | cut -d : -f 2

free
ps -aux
ps -fax
pstree
top
uptime
jobs

apt-get update / apt-get upgrade
apt-get install appname
dpkg -l / dpkg --search / dpkg --s

ip addr show
env / printenv

chown
chmod / chmod u=rwx,g=rx,o= /home/user /
chmod 750 /home/user

> file.txt
>> file.txt
ls -la /usr/bin &> file.txt
ls -la /usr/bin 2> file.txt

ls -la /usr/bin > file.txt 2>&1
dd if=/dev/zero of=/home/user/f4_4 bs=1024
count=2000

kill -SIGTERM 10492
kill -SIGKILL 10492
killall -SIGKILL processname

lsblk
fdisk -l
mkfs / mkfs.ext4 / mkfs.ntfs
mount
umount
lsdf
cat /proc/partitions
df -h
mkswap
swapon
swapoff
cat /proc/swaps
/etc/fstab
dd if=/dev/zero of=/dev/sdb1 bs=1024
count=10
fsck

/dev/zero
/dev/random
/dev/null

find . -type d -name "example"
find . -type f -iname "example.*"
find . -not -type f -iname "example.*"
find . -name "abc*" ! -name "*.php"

```
find . -name "*.php" -o -name "*.txt"
find . -type f -mmin +1 -mmin -5
find . -size +5M
find . -empty
find . -perm 777
find example/ -type d -exec chmod 775 {} +
find . -maxdepth 1 -type f -name "*.jpg"
-exec rm {} +
```

```
tar -cvf example.tar directory/
tar -tf example.tar
tar -xvf example.tar
gzip example.tar
gunzip example.tar
bzip2 example.tar
bunzip2 example.tar
tar -cvzf example.tar.gz directory/
tar -xvzf example.tar.gz
tar -cvjf example.tar.bz2 directory/
tar -xvjf example.tar.bz2
gzip < /directory/example > example.gz
bzip2 < /directory/example > example.bz2
```

sha1sum file

```
sed "s/pattern/newpattern/g" file
sed -i "s/pattern/newpattern/g" file
sed "s/s*#.*/g; /^$/ d; s/^[:space:]*//g" file
sed "s/s*#.*/g; /^$/ p; s/^[:space:]*//g" file
sed "s/s*#.*/g; /^$/ q; s/^[:space:]*//g" file
sed "10 q" file
```

seq 10

Regular expressions

Format

<code>^</code> and <code>\$</code>	Start / end of a line
<code>.</code>	Any character
<code>[]</code> and <code>[^]</code>	Any character (not) between the brackets
<code>?</code>	Zero or one time previous character / expression
<code>*</code> and <code>+</code>	Zero or more / one time previous character / expression
<code>{x,y}</code>	Minimum x and maximum y previous character / expression
<code>()</code>	Group

Character classes

<code>\w</code> and <code>\W</code>	“word character” (a-zA-Z_) and inverse
<code>\b</code> and <code>\B</code>	“word boundary” (boundary from a word) and inverse
<code>\s</code> and <code>\S</code>	Whitespace and inverse
<code>[[:alpha:]]</code>	a-zA-Z
<code>[[:digit:]]</code>	0-9

<code>[[:alnum:]]</code>	a-zA-Z0-9
<code>\d</code> and ...	Not in grep: same as <code>[[:digit:]]</code>

Examples

KdG student numbers: `[0-9]{7}-[0-9]{2}`

Hexadecimal number of 4 numbers: `[0-9A-Fa-f]{4}`

Each number containing a minimum of 3 zeros, repeated after each other:
`[0-9]*0{3}[0-9]*`

Word “fix” in a text, different possibilities:
`[[:space:]]fix[[:space:]]`
`fix\W`
`\<fix\>`

Start with `<`, contains `@` and ends with `>`:
`<.+@.+>`

Bash shell scripting

```
#!/usr/bin/env bash
```

```
# comments
```

```
var="Hello"
```

```
export globalvar="Hello"
```

```
clear
```

```
echo -n "Enter your name: "
```

```
read name
```

```
echo -e $var,\n$name
```

```
read -p "What is your first name? " firstname
```

```
echo "${firstname^}" / echo "${firstname^^}"
```

Positional parameters

\$0 (0-9)

\$# (get amount of positional parameters)

\$* and \$@ (list of all parameters)

Quotes

Single quotes: hard quotes, print what's between them

Double quotes: soft quotes, \$ and ` will be handled escape with \

Backquotes: command substitution or use \$(..)

Calculate with + - * / %

```
number=$((2+2))
```

```
let number=2+2
```

```
chmod +x script.sh
```

```
./script.sh
```

```
source script.sh
```

```
. script.sh
```

/bin/true (0) - /bin/false (1)

\$? (exit status -- exit 113)

[..] or newer version [[..]] (with regular expressions =~)

-n true if next variable has a value

-z true if the string is empty

-d true if it is a directory

-f true if it is a file

-r true if it is a readable file

-w true if there are writing permissions for the file

-x true if it is an executable file

[file1 -nt file2] true if file1 is newer than file2

[file1 -ot file2] true if file1 is older than file2

-ot reversed

[number1 .. number2]

-lt less than

-le less than or equals

-eq equals

-gt greater than

-ge greater than or equals

-ne not equals

```
[ -d "$1" ] && echo "It's a directory"
```

```
[ -f "$1" ] && echo "It's a file"
```

```
[ -x "$1" ] && { echo "Not allowed with an executable"; exit 1; }
```

```
[ ! -z $EUID ] && { echo "You are not root"; exit 1; }
```

```
[[ "$int" =~ ^-[0-9]+$ ]]
```

```
(( int == 0 )) / (( int < 0 )) / (( (int % 2) == 0 ))
```

```
[[ .. ]] && [[ .. ]] / [[ .. ]] || [[ .. ]] / ! (( int == 0 ))
```

IFS (space / tab / newline) / IFS=\$'\n' (only newline)

```
for filename in $(find ~/ -iname '*.txt')
do
    echo $filename
done
```

```
for filename in $(ls *.tar.gz)
do
    tar xvf $filename
done
```

```
if [ -d /etc/systemd ]; then
    echo "Directory exists"
fi
```

```
if ls ~/tmp/*.tar.gz &> /dev/null; then
    cp ~/tmp/*.tar.gz .
else
    echo "Er zijn geen tar.gz-bestanden beschikbaar"
    exit 2
fi
```

```
if [ -d ~/Music ] && [ -w ~/Music ] && [ -x ~/Music ]; then
    cd ~/Music
elif [ -d ~/Documents ] && [ -w ~/Documents ] && [ -x ~/Documents ]; then
    cd ~/Documents
else
    echo "No access to a directory"
    exit 1
fi
```

```
teller=$1
[ ! -z "$1" ] && { echo "Expecting one parameter"; exit 1 }
while [ $teller -gt 0 ]; do
    echo $teller
    sleep 1
   =$((teller--))
done
```

```
case $1 in
    move)
        echo "Move"
        ;;
    copy | kopie)
        echo "Copy"
        ;;
    delete)
        echo "Delete"
        ;;
    *)
        echo "Something else"
esac
```

```
function name {
    local foo # local variable for this function
    echo "Do something with $foo"
    return
}
```

cmd1 && cmd2 (execute cmd2 if cmd1 exit code 0)
cmd1 || cmd2 (execute cmd2 only if cmd1 exit code 1)