# Cheatography

# Sheeeeeeeeeeet Cheat Sheet

by Spec via cheatography.com/21500/cs/4158/

## vim

| | |
|---|---|
| zo | open folded text |
| zc | close folded text |
| zr | unfold one level |
| zm | fold one level |
| zR | unfold all |
| zM | fold all |
| c-w w | switch window |
| ctrl-w s | split windows |
| c-w v | split vertically |
| c-w q | quit window |
| `:set list` | show characters |
| do | diff obtain |
| dp | diff put |
| `:set diffop -t+= iwhite` | avoid whitespace in diff |
| `:diffu` | refresh differences |

**Auto wrap text:**

1. `:set textwi dth=80`
2. select text in visual mode
3. `gq`

## regex

| | |
|---|---|
| [^bg] | caret negates expression |
| \s | whitespace |
| \S | not whitespace |
| \d | digit |
| \D | not digi |
| \w | word |
| \x | hexadecimal digit |
| \O | octal digit |
| (?:...) | Passive (non-capturing) group |

## screen

| | |
|---|---|
| C-a S | split horizontally |
| C-a \| or C-a V | split vertically |
| C-a tab | jump to next region |
| C-a X | close region |

## Syslog severity levels

| | | |
|---|---|---|
| 0 | emerg | Emergency |
| 1 | alert | Alert |
| 2 | crit | Critical |
| 3 | err | Error |
| 4 | warning | Warning |
| 5 | notice | Notice |
| 6 | info | Informational |
| 7 | debug | Debug |

## IO redirection

**Output of cmd2 as file input to cmd1:**

`cmd1 <(cmd2)`

**Stderr of cmd to file:**

`cmd 2> file`

**Stdout to same place as stderr:**

`cmd 1>&2`

**All output of cmd to file:**

`cmd &> file`

**stderr of cmd1 to cmd2:**

`cmd1 |& cmd2`

## DH Groups

DH Group 1: 768-bit group
DH Group 2: 1024-bit group
DH Group 5: 1536-bit group
DH Group 14: 2048-bit group
DH Group 15: 3072-bit group
DH Group 19: 256-bit elliptic curve group
DH Group 20: 384-bit elliptic curve group

**IPsec VPN:**

UDP 500 (IKE includes ISAKMP)
UDP 4500 (NAT traversal)
IP protocol 50 (ESP)
IP protocol 51 (AH)

## nmap

| | |
|---|---|
| -sn | no port scan |
| -PE | only ICMP echo for host disc. |
| -Pn | no ping |
| -n | no name resolve |
| -p80,443 | scan port 80 and 443 |
| -p- | scan all ports |
| -e *<IF-ACE>* | use specified interface |
| -O | OS detection |

## tcpdump

| | |
|---|---|
| -A | payload in ASCII |
| -c <count> | exit after *count* packets |
| -e | print link level headers |
| -G <n> | rotate dump file every *n* sec |
| -s <len> | snip first *len* bytes per packet |
| -S | absolute TCP seq numbers |

## random

**Backuppaa ACL:t:**

`getfacl -pR /joku/ hak emisto > permis sio ns.acl`

`setfacl --rest ore =pe rmi ssi -ons.acl`

**Listaa ladattavat moduulit:**

`cat /lib/m odu les /$( uname -r)/mo dul es.dep`

**TCP portit:**

`0-1023` - well-known ports

`1024-49151` - registered port

`49152- 65535` - dynamic / private ports

**SSL netcat:**

`openssl s_client -connect localh ost:995`

**Maalaa osumat:**

`grep -E --color '(^|haku)'`

**Unix epoch aika:**

`date +%s`

`date -d @12687 27836`

# Cheatography

# Sheeeeeeeeeeet Cheat Sheet

by Spec via cheatography.com/21500/cs/4158/

## sed: address notation

| | |
|---|---|
| n | line number |
| $ | last line |
| /regexp/ | lines matching the pattern |
| addr1,addr2 | range of lines |
| 10~2 | start `10`, then by `2` intervals |
| addr1,+n | `addr1` and following `n` lines |
| addr! | all lines except `addr` |

sed -n '**1,5**p' file.txt
sed '**1,5**s/old/new' file.txt

## sed: editing commands

| | |
|---|---|
| = | output line number |
| a | append text after line |
| d | delete line |
| p | print line, use -n to override |
| s/old/new/ | substitute |

```
sed "1,10 d" file.txt
sed " /snmp/a text to append " file.txt
sed -n " /snmp/ =" file.txt
sed -i -e '/suod ata /{s /et -si/ kor vaa/}' file
```

## </> Bash: space split

```
INPUT: "word" "spaces yo"
$*    : $1 = word, $2 = spaces, $3 = yo
" $*" : $1 = word spaces yo, $2 =, $3 =
$@    : $1 = word, $2 = spaces, $3 = yo
" $@" : $1 = word, $2 = spaces yo, $3 =
```

## </> Bash: for in C Form

```
for (( i=0; i<5; i=i+1 )); do
   echo $i
done
```

## </> Bash: variable expansions

**Return *$parameter* or "*word*" if empty:**

```
${para met er: -word}
```

**Previous + set variable value if it's unset:**

```
${para met er: =word}
```

**Exit script with *error message* if empty:**

```
${para met er: ?error message}
```

**Results nothing if empty. If not empty, substitute and don't change variable value:**

```
${para met er: +word}
```

**Return variable names:**

```
${!pre fix*}
```

**Return string length:**

```
${#par ameter}
```

**Return substring:**

```
${para met er: offset}
${para met er: off set :le ngth}
```

**Remove leading substring:**

```
${para met er# pat ter n} (shortest match)
${para met er# #pa tter n} (longest match)
```

**Remove trailing substring:**

```
${para met er% pat ter n} (shortest match)
${para met er% %pa tter n} (longest match)
```

**Search and replace. If string not set, delete matched text.**

**Only first occurence:**

```
${para met er/ pat ter n/s -tring}
```

**All occurences:**

```
${para met er/ /pa tte rn/ -string}
```

**Match required at the beginning of string:**

```
${para met er/ #pa tte rn/ -string}
```

**Match required at the end of string:**

```
${para met er/ %pa tte rn/ -string}
```

## </> Bash: number bases

| | |
|---|---|
| **number** | base 10 integer |
| 0**number** | base 8 octal |
| 0x**number** | base 16 hexadecimal |
| **base#number** | *number* is in *base* |

```
echo $((0xff))
```

▸ 255

```
echo $((2#1 111 1111))
```

▸ 255

## </> Bash: ternary operator

```
expr1?expr2:expr3
```

If expression ***expr1*** evaluates to be non-zero (arithmetic true) then ***expr2***, else ***expr3***.

## </> Bash: case conversion

| | |
|---|---|
| ${parameter,,} | All to lowercase. |
| ${parameter,} | First char to lowercase. |
| ${parameter^^} | All to uppercase. |
| ${parameter^} | First char to uppercase. |

## </> Bash: tests

**Pattern matching:**

```
[[ " foo.ba r" == foo.* ]]
```

**Regexp:**

```
[[ " $IN T" =~ ^-?[0-9]+$ ]]
```

**Arithmetic truth tests:**

```
(( ((INT % 2)) == 0))
```