BASH 快速参考

命令、脚本、管道、重定向、作业控制

基础

echo 与导航

```
echo "Hello, World!"     # print text
pwd                      # print working directory
cd /path/to/dir          # change directory
cd ..                    # go up one level
cd ~                     # go to home directory
cd -                     # go to previous directory
```

列出与创建

```
ls              # list files
ls -la          # long format, show hidden
ls -lh          # human-readable sizes
mkdir mydir     # create directory
mkdir -p a/b/c  # create nested directories
```

复制、移动与删除

```
cp file.txt copy.txt      # copy file
cp -r dir/ backup/        # copy directory recursively
mv old.txt new.txt        # rename / move
rm file.txt               # delete file
rm -r dir/                # delete directory recursively
rm -rf dir/               # force delete (no prompt)
```

变量与展开

变量

```
name="Alice"              # assign (no spaces!)
echo "$name"              # variable expansion
echo "${name}_file"       # braces for clarity
readonly PI=3.14          # constant
unset name                # delete variable
```

特殊变量

`$0`	脚本名称
`$1 $2 ...`	位置参数
`$#`	参数个数
`$@`	所有参数（各自独立）
`$*`	所有参数（单个字符串）
`$?`	上条命令的退出状态
`$$`	当前进程 ID
`$!`	最后一个后台进程的 PID

命令替换与算术

```
files=$(ls)               # capture output
today=$(date +%Y-%m-%d)   # command substitution
count=$((5 + 3))          # arithmetic: 8
echo $((10 / 3))          # integer division: 3
echo $((10 % 3))          # modulo: 1
```

字符串操作

`${#str}`	字符串长度
`${str:0:5}`	子串（偏移量：长度）
`${str/old/new}`	替换第一个匹配
`${str//old/new}`	替换所有匹配
`${str^^}`	转大写
`${str,,}`	转小写

条件判断

if / elif / else

```
if [[ "$name" == "Alice" ]]; then
    echo "Hi Alice"
elif [[ "$name" == "Bob" ]]; then
    echo "Hi Bob"
else
    echo "Who are you?"
fi
```

测试运算符

`-eq  -ne`	整数等于 / 不等于
`-lt  -gt`	整数小于 / 大于
`-le  -ge`	整数小于等于 / 大于等于
`==  !=`	字符串等于 / 不等于
`-z "$str"`	字符串为空
`-n "$str"`	字符串非空
`-f file`	文件存在且为普通文件
`-d dir`	目录存在
`-e path`	路径存在（任意类型）
`-r  -w  -x`	可读 / 可写 / 可执行
`&&  ||`	逻辑与 / 或

循环

for 循环

```
for fruit in apple banana cherry; do
    echo "$fruit"
done

for f in *.txt; do
    echo "File: $f"
done
```

C 风格 for 循环

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

while 循环

```
count=0
while [[ $count -lt 5 ]]; do
    echo "$count"
    ((count++))
done
```

循环控制

`break`	退出循环
`continue`	跳到下一次迭代

函数

定义与调用

```
greet() {
    echo "Hello, $1!"   # $1 = first arg
    return 0            # exit status
}
greet "Alice"           # Hello, Alice!
```

局部变量与返回值

```
add() {
    local sum=$(($1 + $2))
    echo "$sum"         # "return" via stdout
}
result=$(add 3 5)       # capture: 8
```

管道与重定向

管道

```
ls -l | grep ".txt"     # pipe output
cat log | sort | uniq   # chain commands
cmd1 | tee out.txt      # pipe + save to file
```

重定向

`cmd > file`	重定向 stdout（覆盖）
`cmd >> file`	重定向 stdout（追加）
`cmd < file`	从文件重定向 stdin
`cmd 2> file`	重定向 stderr
`cmd 2>&1`	将 stderr 重定向到 stdout
`cmd &> file`	重定向 stdout + stderr
`cmd << EOF`	Here document（内联输入）
`/dev/null`	丢弃输出：`cmd > /dev/null`

文件操作

查看文件

```
cat file.txt              # print entire file
head -n 10 file.txt       # first 10 lines
tail -n 10 file.txt       # last 10 lines
tail -f log.txt           # follow (live updates)
less file.txt             # paginated viewer
```

统计与查找

```
wc -l file.txt            # count lines
wc -w file.txt            # count words
wc -c file.txt            # count bytes
find . -name "*.txt"      # find by name
find . -type d            # find directories
find . -mtime -7          # modified in last 7 days
```

其他文件命令

`touch file`	创建文件 / 更新时间戳
`stat file`	文件元数据（大小、日期）
`file img.png`	检测文件类型
`diff a.txt b.txt`	对比两个文件
`sort file.txt`	按行排序
`uniq`	去除相邻重复行
`cut -d: -f1`	按分隔符提取字段
`tr 'a-z' 'A-Z'`	字符转换 / 替换

文本处理

grep

```
grep "error" log.txt       # search for pattern
grep -i "error" log.txt    # case-insensitive
grep -r "TODO" src/        # recursive search
grep -n "func" file.go     # show line numbers
grep -c "error" log.txt    # count matches
grep -v "debug" log.txt    # invert match
```

sed

```
sed 's/old/new/' file      # replace first per line
sed 's/old/new/g' file     # replace all
sed -i 's/old/new/g' file  # edit in place
sed -n '5,10p' file        # print lines 5-10
sed '/pattern/d' file      # delete matching lines
```

awk

```
awk '{print $1}' file      # print first field
awk -F: '{print $1}' file  # custom delimiter
awk '$3 > 100' file        # filter by field value
awk '{sum+=$1} END{print sum}' file  # sum column
```

权限

chmod

```
chmod 755 script.sh    # rwxr-xr-x
chmod +x script.sh     # add execute
chmod -w file.txt      # remove write
chmod u+x,g-w file     # user +exec, group -write
```

权限参考

`r (4)`	读
`w (2)`	写
`x (1)`	执行
`u / g / o`	所有者 / 组 / 其他
`755`	所有者：rwx，组/其他：r-x
`644`	所有者：rw-，组/其他：r--

所有权

```
chown user file.txt          # change owner
chown user:group file.txt    # change owner + group
chown -R user:group dir/     # recursive
```