BASH クイックリファレンス

コマンド、スクリプト、パイプ、リダイレクト、ジョブ制御

基本操作

echo & ナビゲーション

```
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
```

一覧表示 & 作成

```
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
```

コピー、移動 & 削除

```
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)
```

変数 & 展開

変数

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

特殊変数

`$0`	スクリプト名
`$1 $2 ...`	位置引数
`$#`	引数の数
`$@`	全引数（個別の単語）
`$*`	全引数（単一の文字列）
`$?`	直前のコマンドの終了ステータス
`$$`	現在のプロセス ID
`$!`	最後のバックグラウンドプロセスの PID

コマンド置換 & 算術

```
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
```

文字列操作

`${#str}`	文字列の長さ
`${str:0:5}`	部分文字列（オフセット：長さ）
`${str/old/new}`	最初のマッチを置換
`${str//old/new}`	全マッチを置換
`${str^^}`	大文字に変換
`${str,,}`	小文字に変換

条件分岐

if / elif / else

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

テスト演算子

`-eq  -ne`	整数の等値 / 非等値
`-lt  -gt`	整数の小なり / 大なり
`-le  -ge`	整数の以下 / 以上
`==  !=`	文字列の等値 / 非等値
`-z "$str"`	文字列が空
`-n "$str"`	文字列が空でない
`-f file`	ファイルが存在し通常ファイル
`-d dir`	ディレクトリが存在
`-e path`	パスが存在（任意の種別）
`-r  -w  -x`	読み取り可 / 書き込み可 / 実行可
`&&  ||`	論理 AND / OR

ループ

for ループ

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

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

C スタイル for ループ

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

while ループ

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

ループ制御

`break`	ループを終了
`continue`	次のイテレーションへスキップ

関数

定義 & 呼び出し

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

ローカル変数 & 戻り値

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

パイプ & リダイレクト

パイプ

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

リダイレクト

`cmd > file`	標準出力をリダイレクト（上書き）
`cmd >> file`	標準出力をリダイレクト（追記）
`cmd < file`	ファイルから標準入力
`cmd 2> file`	標準エラーをリダイレクト
`cmd 2>&1`	標準エラーを標準出力へ
`cmd &> file`	標準出力と標準エラーをリダイレクト
`cmd << EOF`	ヒアドキュメント（インライン入力）
`/dev/null`	出力を破棄: `cmd > /dev/null`

ファイル操作

ファイルの表示

```
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
```

カウント & 検索

```
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
```

その他のファイルコマンド

`touch file`	ファイル作成 / タイムスタンプ更新
`stat file`	ファイルのメタデータ（サイズ、日時）
`file img.png`	ファイルタイプを判定
`diff a.txt b.txt`	2 ファイルの差分比較
`sort file.txt`	行をソート
`uniq`	隣接する重複を削除
`cut -d: -f1`	区切り文字でフィールド抽出
`tr 'a-z' 'A-Z'`	文字の変換 / 置換

テキスト処理

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
```

パーミッション

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
```

パーミッション早見表

`r (4)`	読み取り
`w (2)`	書き込み
`x (1)`	実行
`u / g / o`	ユーザー / グループ / その他
`755`	オーナー: rwx、グループ/その他: r-x
`644`	オーナー: rw-、グループ/その他: r--

所有者変更

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