Bash 치트시트

1. 기본 및 변수

- `echo <string>`: 문자열 출력. `-e` 옵션으로 이스케이프 시퀀스(n, t) 사용 가능.
- `printf <format> <args>`: 형식화된 출력. (예: `printf "Name: %s\nID: %d\n" "Jules" 123`)
- `read <var>`: 사용자 입력 읽기. `-p "Prompt"` 프롬프트 지정, `-s` 비밀번호처럼 숨김.
- 변수 선언: `variable="value"` (등호 주위에 공백 없음)
- 변수 사용: `$variable` 또는 `${variable}`
- 명령어 치환: `output=$(command)` 또는 `command`
- 환경 변수: `export VAR="value"`
- 특수 변수:
  - `$0`: 스크립트 이름
  - `$1`, `$2`…: 위치 매개변수
  - `$#`: 인수의 개수
  - `$*`: 모든 인수를 하나의 문자열로 취급
  - `$@`: 모든 인수를 개별 문자열로 취급 (따옴표와 함께 사용할 때 중요)
  - `$?`: 마지막 명령어의 종료 코드 (0은 성공, 0이 아니면 실패)
  - `$$`: 현재 셸의 프로세스 ID (PID)

2. 조건문

- `if [ condition ]; then ... fi`: 기본 if 문
- `if [[ condition ]]; then ... fi`: 확장된 조건문 (정규 표현식 등 지원, 더 안전)
- `case $variable in pattern1) ... ;; pattern2) ... ;; *) ... ;; esac`: case 문

조건 표현식 (`[[ ... ]]` 내부)

- 문자열: `[[ "$a" == "$b" ]]`, `[[ "$a" != "$b" ]]`, `[[ -z "$a" ]]` (zero length), `[[ -n "$a" ]]` (not-zero length)
- 정규 표현식: `[[ "$str" =~ $regex ]]`
- 숫자: `(( a == b ))`, `(( a > b ))` 등. 또는 `[ $a -eq $b ]`, `[ $a -gt $b ]` (구식)
- 파일:
  - `-e`: 파일/디렉토리 존재 여부
  - `-f`: 일반 파일 여부
  - `-d`: 디렉토리 여부
  - `-r`, `-w`, `-x`: 읽기/쓰기/실행 가능 여부
- 논리: `&&` (AND), `||` (OR), `!` (NOT)

3. 반복문

- `for var in item1 item2 ...; do ... done`
- `for var in {1..10}; do ... done` (Brace Expansion)
- `for var in $(command); do ... done`
- C-style for loop: `for (( i=0; i<10; i++ )); do ... done`
- `while condition; do ... done`
  - `while read -r line; do ... done < file.txt`: 파일을 한 줄씩 읽기
- `until condition; do ... done`
- `break`: 루프 종료
- `continue`: 다음 반복으로 이동

4. 함수

- 정의 (두 가지 방법):

```
function my_func { ... }
my_func() { ... }
```

- 호출: `my_func arg1 arg2`
- 인수 접근: `$1`, `$2`, `$@` 등 스크립트와 동일
- `return <number>`: 숫자 종료 코드(0-255) 반환.
- 함수 출력 캡처: `result=$(my_func)`
- `local <var>`: 함수 내에서 지역 변수 선언.

5. 배열 (Arrays)

- 선언: `arr=("apple" "banana" "cherry")`
- 요소 접근: `${arr[0]}` (첫 번째 요소)
- 모든 요소: `${arr[@]}`
- 모든 인덱스: `${!arr[@]}`
- 배열 길이: `${#arr[@]}`
- 요소 추가: `arr+=("new item")`
- 연관 배열 (Associative Arrays, key-value):
  - `declare -A my_map`
  - `my_map["key1"]="value1"`
  - `echo ${my_map["key1"]}`

6. 문자열 조작

- 길이: `${#string}`
- 부분 문자열: `${string:position:length}` (예: `${string:0:5}`)
- 패턴 제거:
  - `${string#pattern}`: 앞에서 가장 짧은 일치 제거
  - `${string##pattern}`: 앞에서 가장 긴 일치 제거
  - `${string%pattern}`: 뒤에서 가장 짧은 일치 제거
  - `${string%%pattern}`: 뒤에서 가장 긴 일치 제거
- 치환:
  - `${string/pattern/replacement}`: 첫 번째 일치 치환
  - `${string//pattern/replacement}`: 모든 일치 치환

7. 셸 확장 (Expansions)

- `~`: 홈 디렉토리
- `*`, `?`, `[...]`: 경로명 확장 (Globbing)
- `{a,b,c}`: Brace Expansion (예: `touch file_{1..3}.txt`)
- `$(...)`: 명령어 치환
- `$((...))`: 산술 확장

8. 스크립팅 팁

- `set -e`: 명령어가 실패하면 즉시 스크립트 종료.
- `set -u`: 정의되지 않은 변수를 사용하면 에러 발생.
- `set -o pipefail`: 파이프라인의 명령어 중 하나라도 실패하면 전체를 실패로 간주.
- `trap 'cleanup' EXIT`: 스크립트 종료 시 `cleanup` 함수 실행.
- `getopts`: 스크립트 옵션(플래그)을 파싱.
- `readlink -f "$0"`: 스크립트의 실제 경로를 얻기.
- `shebang`: 스크립트 첫 줄에 `#!/bin/bash` 또는 `#!/usr/bin/env bash`를 명시하여 실행할 인터프리터를 지정.