Command Substitution

Allows the output of a command to be used as input in another command. It enables dynamic command execution within scripts and command lines.

Syntax

There are two ways to perform command substitution:

1. **Backticks (``)** (Old-style)

   ```
   output=`command`
   ```

2. **()$Syntax(Preferred) * *“‘bashoutput =$(command) “‘

Ex: Nesting Commands

```
# Using backticks (harder to read)
date_info=`echo "Today is \`date\`"`

# Using $() (easier to read)
date_info=$(echo "Today is $(date)")
```

Ex 1: Storing Command Output in a Variable

```
current_date=$(date)
echo "Current date and time: $current_date"
```

Ex 2: Using Command Substitution in a Loop

```
for file in $(ls *.txt); do
    echo "Processing file: $file"
done
```

NOTE

- Command substitution runs in a **subshell**, which may have performance overhead.
- Avoid unnecessary subshells in loops for efficiency.

Ex: Avoiding Unnecessary Subshells in Loops

Inefficient

```
for item in $(ls); do
    echo "$item"
done
```

Efficient (Using `find` and `while` Loop)

```
find . -maxdepth 1 -type f | while read -r file; do
    echo "$file"
done
```