Notes 9

grep

Definition

- `grep` is used to search for specific text patterns within files or command output.

Usage / Formula

```
 grep [options] "pattern" file
```

Examples

```
grep "error" system.log
```

Searches for the word `error` in `system.log`.

```
grep -i "login" access.log
```

Searches for `login` in `access.log`, ignoring case.

```
grep -n "root" /etc/passwd
```

Displays matching lines and their line numbers.

---

awk

Definition

- `awk` is a powerful text-processing language used for pattern scanning and data extraction, especially for structured text like columns.

Usage / Formula

```
awk 'pattern { action }' file
```

Examples

```
awk '{print $1}' data.txt
```

Prints the first column of each line in `data.txt`.

```
awk '/SUCCESS/ {print $3, $4}' access.log
```

Prints the 3rd and 4th fields from lines containing `SUCCESS`.

```
awk -F":" '{print $1}' /etc/passwd
```

Uses `:` as a delimiter and prints usernames.

---

sed

Definition

- `sed` is used to perform basic text transformations such as search, replace, insert, and delete operations.

Usage / Formula

```
sed 'command' file
```

Examples

```
sed 's/error/warning/' file.txt
```

Replaces the first occurrence of `error` with `warning` on each line.

```
sed 's/error/warning/g' file.txt
```

Replaces all occurrences of `error` with `warning`.

```
sed '1d' file.txt
```

Deletes the first line of `file.txt`.

Pipe Operator |

Definition

- The pipe (`|`) redirects the output of one command directly into another command as input.

Usage / Formula

```
command1 | command2
```

Examples

```
cat access.log | grep "ERROR"
```

Sends the contents of `access.log` to `grep` to filter lines with `ERROR`.

```
ps aux | grep root
```

Searches running processes owned by `root`.

```
ls -l | wc -l
```

Counts the number of files in the directory.

Output Redirection > (Save Output to a File)

Definition

- The `>` operator redirects command output to a file, **overwriting** the file if it already exists.

Usage / Formula

```
command > file
```

Examples

```
ls > files.txt
```

Saves the directory listing to `files.txt`.

```
date > today.txt
```

Writes the current date to `today.txt`.

```
grep "ERROR" log.txt > errors.txt
```

Stores all `ERROR` lines into `errors.txt`.

Output Redirection >> (Append Output to a File)

Definition

The `>>` operator appends command output to the end of a file without overwriting existing content.

Usage / Formula

```
command >> file
```

Examples

```
echo "New entry" >> notes.txt
```

Adds `New entry` to `notes.txt`.

```
date >> log.txt
```

Appends the current date to `log.txt`.

```
grep "FAIL" access.log >> failures.txt
```

Adds failed access attempts to `failures.txt`.
