Linux Command Notes

1. `grep`

Definition

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

Usage / Formula

```
grep [options] "pattern" filename
```

Examples

```
grep "error" logfile.txt
```

```
grep -i "warning" /var/log/syslog
```

```
dmesg | grep usb
```

---

2. `awk`

Definition

`awk` is a powerful text-processing language used for pattern scanning, data extraction, reporting, and transforming text files.

Usage / Formula

```
awk 'pattern { action }' filename
```

Examples

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

```
awk '/error/ {print $0}' logfile.txt
```

```
ps aux | awk '{print $1, $2, $11}'
```

---

3. sed

Definition

`sed` (stream editor) is used for editing text in a pipeline, performing search-and-replace, insertion, deletion, and more.

Usage / Formula

```
sed 's/search/replace/' filename
```

Examples

```
sed 's/apple/orange/' fruits.txt
```

```
sed -i 's/localhost/127.0.0.1/' config.txt
```

```
echo "hello world" | sed 's/world/Linux/'
```

Using the Pipe |

Definition

A pipe sends the output of one command as the input to another command.

Examples

```
ls -l | grep ".txt"
```

```
ps aux | awk '{print $1, $2}'
```

```
cat access.log | wc -l
```

Saving Output to a File using >

Definition

The > operator redirects command output and *overwrites* the target file.

Examples

```
ls > files.txt
```

```
echo "Backup completed" > status.log
```

```
grep "error" system.log > errors_only.txt
```

Appending Output to a File using >>

Definition

The >> operator *adds* the output of a command to the end of a file without overwriting it.

Examples

```
date >> backup.log
```

```
who >> active_users.txt
```

```
echo "Process finished" >> script_output.txt
```