Notes 7: Linux Text Processing Commands

This document explains how to use common Linux text processing commands with definitions, syntax, and examples.

`cat` - Concatenate and Display Files

Definition:
`cat` (short for "concatenate") is used to display the contents of a file or concatenate multiple files together.

Usage:

```
cat [OPTION]... [FILE]...
```

Examples:

```
cat file.txt
cat file1.txt file2.txt > merged.txt
cat -n file.txt  # show line numbers
```

`tac` - Reverse Concatenate

Definition:
`tac` displays file contents in reverse order (line-wise), opposite of `cat`.

Usage:

```
tac [FILE]
```

Examples:

```
tac file.txt
tac file1.txt > reversed.txt
tac log.txt | head -n 10
```

`head` - Output the First Part of Files

Definition:
`head` shows the first lines of a file (default is 10).

Usage:

```
head [OPTION]... [FILE]
```

Examples:

```
head file.txt
head -n 5 file.txt
head -c 20 file.txt  # first 20 bytes
```

`tail` - Output the Last Part of Files

Definition:
`tail` shows the last lines of a file (default is 10).

Usage:

```
tail [OPTION]... [FILE]
```

Examples:

```
tail file.txt
tail -n 15 file.txt
tail -f /var/log/syslog  # follow log in real time
```

`cut` - Remove Sections from Each Line

Definition:
`cut` removes sections from each line of input based on delimiter or character position.

Usage:

```
cut OPTION... [FILE]...
```

Examples:

```
cut -d "," -f1,2 data.csv
cut -c1-5 file.txt
cut -d ":" -f1 /etc/passwd
```

`sort` - Sort Lines of Text

Definition:
`sort` orders lines in a file or from input.

Usage:

```
sort [OPTION]... [FILE]...
```

Examples:

```
sort names.txt
sort -r numbers.txt  # reverse order
sort -n data.txt     # numeric sort
```

`wc` - Word, Line, Character, and Byte Count

Definition:
`wc` displays counts of lines, words, bytes, and characters.

Usage:

```
wc [OPTION]... [FILE]...
```

Examples:

```
wc file.txt
wc -l file.txt
wc -w file.txt
```

`tr` - Translate or Delete Characters

Definition:
`tr` is used to translate, squeeze, or delete characters from input.

Usage:

```
tr [OPTION]... SET1 [SET2]
```

Examples:

```
echo "hello" | tr a-z A-Z
echo "122333" | tr -s '3'
echo "hello" | tr -d 'aeiou'
```

`diff` - Compare Files Line by Line

Definition:
`diff` compares files line by line and shows differences.

Usage:

```
diff [OPTION] FILE1 FILE2
```

Examples:

```
diff old.txt new.txt
diff -y file1.txt file2.txt  # side-by-side
diff -u file1.txt file2.txt  # unified diff
```

`grep` - Search Text Using Patterns

Definition:
`grep` searches for patterns in files using regular expressions.

Usage:

```
grep [OPTIONS] PATTERN [FILE...]
```

Examples:

```
grep "error" log.txt
grep -i "warning" system.log
grep -r "config" /etc/
```