Redirection and Pipes

This handout covers essential Linux techniques for handling inputs, outputs, errors, combining commands, and processing text.

---

1. Standard Streams in Linux

Linux uses three primary communication channels:

- **Standard Input (*stdin*)**: Data input (`stream 0`)
- **Standard Output (*stdout*)**: Normal program output (`stream 1`)
- **Standard Error (*stderr*)**: Error messages (`stream 2`)

---

2. Output Redirection

Redirect command output (`stdout`) into a file.

- **Overwrite existing file or create new:**

```
echo "Hello, Linux!" > file.txt
```

- **Append output to file:**

```
ls >> file.txt
```

Redirecting Error Output (`stderr`)

Redirect errors to a separate file:

```
du missingfile.txt 2> errors.txt
```

Redirecting both `stdout` and `stderr`

Separately:

```
du file.txt missingfile.txt > output.txt 2> error.txt
```

Together (same file):

```
du file.txt missingfile.txt > combined.txt 2>&1
```

> ***Important: Order matters! 2>&1 must come after redirecting stdout.***

---

3. Standard Input Redirection

Redirect input from a file into a program:

```
wc -l < file.txt
```

It's usually simpler to pass files directly:

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

---

4. Pipes (`|`)

Pipes connect the output of one command directly into another command's input.

Basic syntax:

```
command1 | command2
```

Examples:

Counting files in the current directory:

```
ls | wc -l
```

Redirecting errors in a pipe:

```
du file.txt missing.txt 2>&1 | wc -l
```

---

5. The `tee` Command

`tee` outputs data simultaneously to terminal and file.

Basic usage:

```
ls | tee file.txt
```

Append mode:

```
ls | tee -a file.txt
```

Useful for logging command outputs (e.g., system updates):

```
sudo apt update | tee update_log.txt
```

---

6. Filtering Data with `grep`

`grep` searches for text patterns in files or streams.

- **Simple fixed-string search:**

```
grep -F "error" logfile.txt
```

- **Search in compressed files:**

```
zcat logfile.gz | grep -F "error"
```

---

7. Extracting Data with `cut`

Extract specific parts from text lines.

- **By byte position:**

```
uptime | cut -b 1-10
```

- **By character position:**

```
cut -c 1-10 file.txt
```

- **By fields (delimited data):**

```
uptime | cut -d' ' -f2
```

---

8. Sorting and Removing Duplicate Lines (`sort` and `uniq`)

- **Sort lines alphabetically or numerically**:

```
sort file.txt
```

- **Remove duplicate lines** (after sorting):

```
sort file.txt | uniq
```

> *Note:*
>
> *`uniq` removes **only consecutive duplicate lines**. **Always sort before** applying `uniq` for complete deduplication.*

- **Count occurrences of unique lines**:

```
sort file.txt | uniq -c
```

Explanation:

- `uniq` is commonly used after `sort` to remove duplicate entries.
- The `c` flag counts how many times each unique line occurs.

9. Text Replacement with `sed`

The `sed` command allows powerful text manipulation and replacements.

- **Replace first occurrence per line:**

```
sed 's/old/new/' file.txt
```

- **Replace all occurrences (global replacement):**

```
sed 's/old/new/g' file.txt
```

- **Using `sed` in pipes:**

```
echo "Hello wrld!" | sed 's/wrld/world/'
```

📌 Summary of Key Commands

Command	Description
`command > file.txt`	Redirect *stdout* to file (overwrite)
`command >> file.txt`	Append *stdout* to file
`command 2> error.txt`	Redirect *stderr* to file
`command > out.txt 2> err.txt`	Redirect *stdout/stderr* separately
`command > file.txt 2>&1`	Redirect *stdout* and *stderr* to the same file
`tee file.txt`	Output to terminal and file simultaneously
`grep -F "text" file.txt`	Search file for fixed-string "text"
`cut -b 1-5`	Extract first 5 bytes of each line
`sort file.txt`	Sort contents alphabetically
`uniq -c`	Count occurrences of unique lines
`sed 's/old/new/g' file.txt`	Replace all occurrences of "old" with "new"

⚠️ Best Practices

- **Always use caution with redirection**, as incorrect commands can overwrite files.
- Regularly use `tee` for critical outputs or logs.
- Pipes (`|`) greatly simplify chaining and filtering operations.
- Use backups or version control when processing important files.