1. Input Redirection (<)

Input redirection means taking input for a command from a file instead of the keyboard.

Example:

```
sort < names.txt
```

- Here, instead of typing input manually, the `sort` command reads from `names.txt`.
- If `names.txt` contains:

  ```
  Zara
  Alice
  John
  ```

  The output will be:

  ```
  Alice
  John
  Zara
  ```

---

2. Error Redirection (2> and 2>>)

Error redirection means sending error messages (stderr) to a file instead of the terminal.

Example:

```
ls /no/such/dir 2> error.log
```

- Normally, this would print:

  ```
  ls: cannot access '/no/such/dir': No such file or directory
  ```

- But with `2> error.log`, the error is stored inside `error.log`.
- The terminal stays clean.

You can also append errors using:

```
ls /no/such/dir 2>> error.log
```

---

3. Piping (|)

Piping means sending the output of one command as input to another.

Example:

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

- `ls -l` lists files in detail.
- `grep ".txt"` filters the list to only show files ending with `.txt`.

Another example:

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

- Counts the number of lines in `access.log`.

Perfect 👍 Let's add output redirection to the list so you see the full picture of shell redirection & piping.

---

4. Output Redirection (> and >>)

Output redirection means sending the standard output (stdout) of a command to a file instead of the screen.

Example 1: Overwrite output

```
ls > files.txt
```

- Instead of showing the list of files on the terminal, it saves them in `files.txt`.
- If `files.txt` already exists, it will be overwritten.

Example 2: Append output

```
ls >> files.txt
```

- This will append the output to `files.txt` instead of overwriting.

🔑 Complete Comparison Table

Feature	Symbol(s)	Stream affected	Purpose / Effect	Example
Input Redirection	<	stdin (0)	Take input from a file instead of keyboard	`sort < names.txt`
Output Redirection	>, >>	stdout (1)	Send output to a file (overwrite >, append >>)	`ls > files.txt`
Error Redirection	2>, 2>>	stderr (2)	Send error messages to a file (overwrite/append)	`ls /bad/path 2> errors.log`
Piping	|	stdout → stdin	Send output of one command as input to another	`ls -l | grep ".txt"`

✅ Quick analogy:

- Input redirection (<) = "read from a file instead of my mouth (keyboard)"
- Output redirection (>, >>) = "write to a file instead of my eyes (screen)"
- Error redirection (2>, 2>>) = "log mistakes somewhere else"
- Piping (|) = "pass my output directly into another tool"

---

1. Redirecting stdout and stderr to different files

```
ls /etc /no/such/dir > output.log 2> error.log
```

- `> output.log` → saves normal results (stdout).

- `2> error.log` → saves errors separately (stderr).

👉 `output.log` will have the valid listing of `/etc`, and `error.log` will contain the error about `/no/such/dir`.

---

2. Redirecting stdout and stderr to the same file

```
ls /etc /no/such/dir > combined.log 2>&1
```

- `> combined.log` → sends stdout to the file.
- `2>&1` → sends stderr to the same place as stdout (so both go into `combined.log`).

---

3. Using pipes with redirection

```
ls /etc 2> errors.log | grep ".conf" > configs.txt
```

- `ls /etc` → lists files in `/etc`.
- `2> errors.log` → any errors go into `errors.log`.
- `| grep ".conf"` → filters the list for `.conf` files.
- `> configs.txt` → saves the filtered results.

👉 End result:

- `configs.txt` → contains only `.conf` files.
- `errors.log` → contains any error messages.

---

4. Suppressing errors completely

```
find / -name "*.log" 2>/dev/null
```

- `2>/dev/null` → throws away all errors (like "Permission denied").
- You only see the successful matches.

---

5. Mixing pipes + input redirection

```
sort < names.txt | uniq > sorted_unique.txt
```

- `< names.txt` → feed `names.txt` into `sort`.
- `| uniq` → remove duplicates.
- `> sorted_unique.txt` → save cleaned result.

🔑 Key Takeaway

- You can chain them together to precisely control data flow:
  - `<` → feed input
  - `>` / `>>` → capture or append output
  - `2>` / `2>>` → capture or append errors
  - `|` → connect commands
