Redirection

- By default every command reads input from the terminal (stdin), writes output to the terminal (stdout), and writes errors to the terminal (stderr).
- These three streams are open for every process.

Standard streams

- stdin — standard input (file descriptor 0)
- stdout — standard output (file descriptor 1)
- stderr — standard error (file descriptor 2)

There are Three Types of redirection

Input redirection

- Read input from a file instead of stdin:
- To do input redirection '<' symbol is used

```
command < file    # read input from file
```

Output redirection

- Write stdout to a file instead of the terminal i.e.output will be written into file instead of stdout
- To do output redirection '>' or '>>' symbol is used

```
command > file     # overwrite file
command >> file    # append to file
```

Error redirection

- Write stderr to a file:
- Error will be written into file instead of stderr
- To do output redirection '2>' or '2>>' symbol is used

```
command 2> file    # overwrite file
command 2>> file   # append to file
```

`

`

`

Pipes

- Using pipe, we can redirect output of any command to the input of any other command.
- Two processes are connected using pipe operator (|).
- Two processes runs simultaneously and are automatically rescheduled as data flows between them.
- If you don't use pipes, you must use several steps to do single task.

```
command1 | command2    #output of command1 will be given as input to command 2
```

Example:

```
who | wc    # output of who command is given as input to wc command
cat file.txt | wc   # output of cat command is given as input to wc command
```

`cut`

- Extract Specific Fields or Characters
- The cut command is used to extract columns, fields, or character ranges from each line of a file or input.

```
cut [options] [file]
```

- Common options:
  - `-d` : Specify a delimiter (default is tab)
  - `-f` : Specify fields (columns) to extract
  - `-c` : Specify character positions to extract
  - Examples:
    - `echo "DBDA DESD DAC DMC DITISS" | cut -d " " -f1`
      - Extracts fields 1 using comma (" ") as a delimiter.
    - `echo "DBDA DESD DAC DMC DITISS" | cut -d " " -f2`
    - `echo "DBDA DESD DAC DMC DITISS" | cut -d " " -f5`
    - `echo "DBDA DESD DAC DMC DITISS" | cut -d " " -f1,2`
      - Extracts fields 1 and 2, using comma (" ")as a delimiter.
    - echo "DBDA DESD DAC DMC DITISS" | cut -d " " -f2,3
    - `cut -d ',' -f1,3 file.csv`
      - Extracts fields 1 and 3, using comma (,) as a delimiter.
    - `ls -l -i -s | cut -d " " -f2`
      - Extracts the second field (file size) from the output of ls -l -i -s command.

`

`

`

tr

- The tr (translate) command is used to convert, squeeze, or delete characters from input.
- tr works only with stdin (standard input).
- You must use input redirection or pipes.

```
tr [options] SET1 [SET2]
```

- Common options:
  - `-d` : Delete characters in SET1
  - `-s` : Squeeze repeated characters in SET1
  - Examples:
    - echo "linux" | tr 'a-z' 'A-Z'
      - Convert sting uppercase
    - echo "HELLO" | tr 'A-Z' 'a-z'
      - Convert string lowercase
    - echo "hi there" | tr -s ' '
      - Squeeze Repeated Characters (-s)
    - echo "hello123" | tr -d '0-9'
      - Remove Characters
    - tr 'a-z' 'A-Z' < file.txt
      - Converts all text in a file to uppercase

Shell metacharacters (globbing)

- `*` — zero or more occurrences of any character
  - e.g., `a*` matches `a`, `ab`, `abc`, etc.
- `?` — exactly one occurrence of any character
  - e.g., `a?` matches `ab`, `ac`, but not `a` or `abc`

Regular expressions (regex)

- Find a pattern in text file(s).
- Regular expressions are patterns used to match text.
- A regular expression pattern is composed of simple characters, or a combination of simple and special characters e.g. /abc/, /ab*c/

grep family

- `grep` : GNU Regular Expression Parser
  - Basic wild-card

        - Basic wild-card characters
            - `$` - find at the end of line.
            - `^` - find at the start of line.
            - `*` - zero or more occurrences of previous character
            - `.` - any single character
            - `[ ]` - any single char in give range or set of chars
            - `[^ ]` - any single char not in give range or set of chars
            - `[...]` — any single character in the set or range
            - `[^...]` — any single character not in the set

- `egrep or grep -E` : Extended Grep

    - Basic + Extended wild-card
        - All basic wild-card characters plus:
          - `+` - one or more occurrences of previous character
          - `?` - zero or one occurrence of previous character
          - `{n}` - exactly n occurrences of previous character
          - `{m,}` - at least m occurrences of previous character
          - `{m,n}` - between m and n occurrences of previous character
          - `|` - alternation (one of the groups)
          - `(...)` - grouping characters
          - `(word1|word2)` - match either word1 or word2

- `fgrep or fgrep -F` : Fixed Grep

    - No wild-card

grep usage and common options

```
grep "pattern" filepath
```

Options:

- `-c` : count number of matching lines
- `-v` : invert match (show non-matching lines)
- `-i` : case-insensitive search
- `-w` : match whole words only
- `-R` : recursive search in a directory
- `-n` : show line numbers