Meta Characters

What are Meta Characters?

- **Definition**: Special characters that have a **specific meaning** in shell scripting or regular expressions.
- **Example**: *, |, $, &, etc.

Types of Meta Characters

1. **Search String Meta Characters**
    - Used for pattern matching.
    - Examples: ^, $, [ ], *, \.
2. **Replacement String Meta Characters**
    - Used for text substitution.
    - Examples: &, \n, \.

Common Meta Characters and Their Actions

Meta Character	Action
^	Matches the **beginning of a line**.
$	Matches the **end of a line**.
.	Matches any **single character**.
*	Matches **zero or more occurrences** of the previous character.
+	Matches **one or more occurrences** of the previous character.
?	Matches **zero or one occurrence** of the previous character.
`	`
[abc]	Matches any **single character** inside the brackets.
[^abc]	Matches any character **not** in the brackets.
[a-z]	Matches a **range** of characters (a to z).
\	Treats the next character literally (escapes it).

Search String Meta Characters - Classes

Character classes allow you to match specific types of characters.

Class	Matches
[:alpha:]	Alphabetic characters (a-z, A-Z).
[:digit:]	Digits (0-9).
[:alnum:]	Alphabetic + numeric characters.
[:upper:]	Uppercase letters (A-Z).

Class	Matches
`[:lower:]`	Lowercase letters (a-z).
`[:space:]`	Whitespace (space, tab, newline).
`[:punct:]`	Punctuation symbols.

Replacement String Meta Characters

Meta Character	Action
`&`	Refers to the entire **matched text** during substitution.
`\n`	Refers to the **subgroup** matched, where `n` is 1 to 9.
`\`	Escapes the next character; treats it **literally**.

Protecting Meta Characters

- To prevent meta characters from being interpreted:
    1. Use **single quotes**: `'***'`.
    2. Use a **backslash**: `\*`.
    3. Double quotes also protect meta characters but allow `$`, `\`, and backticks to be processed.

Regular Expressions and `grep`

What is a Regular Expression (regex)?

- A **pattern** used to match strings or text in files or data streams.
- Regular expressions are used in tools like:
    - `grep`, `egrep`, `sed`, `awk`.
    - Programming languages like Python, Perl, and Java.

Types of Regular Expressions

1. **Basic Regular Expression (BRE)**: Used in `grep`.
2. **Extended Regular Expression (ERE)**: Used in `egrep`.

Important Regex Concepts

1. Character Classes

- `[abc]`: Matches any one of `a`, `b`, or `c`.
- `[^abc]`: Matches anything **except** `a`, `b`, or `c`.
- `[a-z]`: Matches any character between `a` and `z`.

Named Classes (POSIX Syntax):

Class	Matches
`[:alpha:]`	Letters (a-z, A-Z).
`[:digit:]`	Digits (0-9).
`[:alnum:]`	Letters and digits.
`[:punct:]`	Punctuation symbols.

---

2. Anchors

- `^`: Matches the **beginning of a line**.
- `$`: Matches the **end of a line**.

Example:

```
grep "^start" file.txt   # Lines starting with "start"
grep "end$" file.txt     # Lines ending with "end"
```

---

3. Repetition Operators

Operator	Meaning
`*`	Zero or more occurrences.
`+`	One or more occurrences.
`?`	Zero or one occurrence.
`{n}`	Exactly `n` occurrences.
`{n,}`	At least `n` occurrences.
`{n,m}`	Between `n` and `m` occurrences.

Example:

```
grep "a\{2,\}" file.txt   # Matches "aa", "aaa", etc.
```

---

4. Subexpressions

- Use `( )` to group parts of a regex.
- `(abc)*`: Matches "abc" zero or more times.

The `grep` Family

Command	Description
`grep`	Searches for patterns using **Basic Regex** (BRE).
`egrep`	Searches for patterns using **Extended Regex** (ERE).

Command	Description
`fgrep`	Searches for **fixed strings** (no regex interpretation).

Common `grep` Options

Option	Description
`-i`	Ignores case distinctions.
`-v`	Inverts the match (show lines not matching).
`-n`	Displays line numbers along with matches.
`-l`	Lists filenames containing the match.
`-e`	Allows specifying multiple patterns.

Example:

```
grep -i "pattern" file.txt     # Case-insensitive search
grep -v "pattern" file.txt     # Show lines that do NOT match
grep -n "pattern" file.txt     # Show line numbers
```

Practical Examples

1. **Find lines starting with "abc"**:

   ```
   grep "^abc" file.txt
   ```

2. **Find lines ending with "xyz"**:

   ```
   grep "xyz$" file.txt
   ```

3. **Find all digits**:

   ```
   grep "[[:digit:]]" file.txt
   ```

4. **Search for multiple patterns**:

   ```
   grep -e "pattern1" -e "pattern2" file.txt
   ```

5. **Find words with exactly two a’s**:

   ```
   grep "a.*a" file.txt
   ```