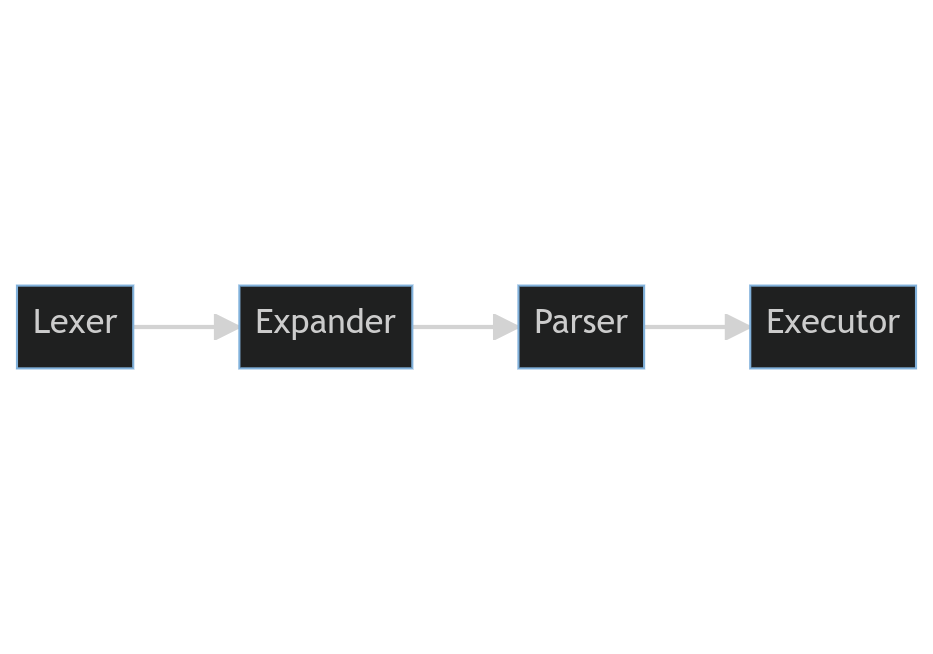
We can divide :

* Lexing / parsing
* Execution

Command parsing, environnement variable ?

how shells generally work



You probably want to structure your minishell like that:

Input -lexer-> tokens  
tokens -parser-> AST  
AST -evaluator-> evaluated AST  
evaluated AST -executor-> output

1. Find out what a *lexer* is. What should it do in the case of minishell? What would be the resulting tokens?
2. Find out what a *parser* is. What could/should it do?
3. What is an AST (*abstract syntax tree*)?  
   What could it look like in our case? Draft programs (+ their AST) on paper.  
   Can you think of an easier way than a tree? If so, make a draft of that, too.

Lexer – need bash tokenization rule

Parser – need bash grammar

As for the parser's output, in case you're not doing the bonus, the way to go might be to use a *Command Table* as shown [here](https://www.cs.purdue.edu/homes/grr/SystemsProgrammingBook/Book/Chapter5-WritingYourOwnShell.pdf).

For both of those: Beware of the Heredoc. (Look [HERE](https://linuxize.com/post/bash-heredoc/).)

Shell functionment

Shell breaks the input into tokens: words and operations

Parses the input into simple command and compound commands

Performs various expansion on different part of each commands

Sure, these terms are all related to how a shell program works and processes user input. Here's a breakdown of each term:

* **Term**: This is the individual unit of user input, typically a word or a command.
* **Parsing**: This is the process of breaking down the user input (tokens) into a structured format that the shell can understand. It's like parsing a sentence into its grammatical components (subject, verb, object).
* **Expander**: This expands variables within the user input. Variables are placeholders that can hold values. The expander replaces these placeholders with the actual values.
* **Lexer**: This is the component that breaks down the user input into individual tokens. It's like breaking a sentence into individual words.
* **Executor**: This is the part of the shell that carries out the commands after they have been parsed. It takes the parsed instructions and executes them.
* **AST (Abstract Syntax Tree)**: This is a tree-like data structure that represents the parsed command. It shows the relationships between different parts of the command.
* **Heredoc**: This is a special way of providing multi-line input within the shell. It allows you to provide a block of text without needing to escape newline characters.

In simpler terms, imagine you typed a command like ls -l | grep file.txt. Here's what happens:

1. **Lexer**: Breaks the command into tokens: ls, -l, |, grep, file.txt.
2. **Expander**: Checks for any variables and expands them.
3. **Parser**: Takes the tokens and builds an AST representing the structure of the command (pipe between two commands).
4. **Executor**: Executes the commands based on the AST. It runs ls -l first, then pipes the output to grep file.txt.
5. **Explication des termes clés d'un Shell**
6. **Jeton**: Un jeton est l'unité individuelle de la saisie de l'utilisateur, généralement un mot ou une commande. Par exemple, dans la commande "ls -l", "ls" et "-l" sont des jetons.
7. **Analyse**: L'analyse est le processus de décomposition de l'entrée de l'utilisateur (jetons) en un format structuré que le shell peut comprendre. C'est comme analyser une phrase en ses composants grammaticaux (sujet, verbe, objet).
8. **Expander**: L'expander développe les variables dans l'entrée de l'utilisateur. Les variables sont des espaces réservés qui peuvent contenir des valeurs. L'expander remplace ces espaces réservés par les valeurs réelles. Par exemple, si la variable "nom" est définie sur "Jean", l'expander remplacera "Bonjour $nom" par "Bonjour Jean".
9. **Lexeur**: Le lexer est le composant qui décompose l'entrée de l'utilisateur en jetons individuels. C'est comme décomposer une phrase en mots individuels. Par exemple, le lexer pour la commande "ls -l" renverrait les jetons "ls", "-", "l".
10. **Exécuteur**: C'est la partie du shell qui exécute les commandes après qu'elles ont été analysées. Il prend les instructions analysées et les exécute. Par exemple, l'exécuteur pour la commande "ls -l" exécutera la commande ls -l et affichera la liste des fichiers et des répertoires du répertoire actuel.
11. **AST (Arbre Syntaxique Abstrait)**: Il s'agit d'une structure de données arborescente qui représente la commande analysée. Il montre les relations entre les différentes parties de la commande. Par exemple, l'AST pour la commande "ls -l | grep fichier.txt" montrerait que la commande "grep fichier.txt" est exécutée sur la sortie de la commande "ls -l".
12. **Heredoc**: Il s'agit d'une manière spéciale de fournir une entrée sur plusieurs lignes dans le shell. Il vous permet de fournir un bloc de texte sans avoir à échapper les caractères de saut de ligne. Par exemple, le code suivant utilise un heredoc pour imprimer un message :
13. cat << END
14. Ceci est un message
15. sur plusieurs lignes.
16. END
17. Ceci imprimera le texte suivant :
18. Ceci est un message
19. sur plusieurs lignes.
20. J'espère que cette explication plus détaillée en français vous a été utile. N'hésitez pas à me poser d'autres questions.