# Web Spider

## *Description :*

This program is a model of a search engine.

## *Idea of the Design :*

The web is modelled as a graph consisting of graph elements in which the web pages are represented by the graph elements.

Each web page consits of links to other web pages in it and the body of the web page. In order to reduce the complexity we are considering only the text part ot the web page.

The graph element stores the data of webpage in three parts:

- Node - Stores the URL of the page
- Children - Stores all the links present in the page
- Contents - Stores the body of the page

Our program consists of two strategies namely

- Search Strategy
- Indexing strategy.

Search Strategy is used for finding a particular web page from the graph created. The Search Strategy may be done in two ways :

- Depth First Strategy
- Breadth First Strategy

Assuming the web in the form of a tree, after searching a node we have a choice of searching a node in its same level or the next level. The choice we make give us one the above strategies.

Depth First Strategy :

In this type of search after the node we proceed to it's children and hence depth first.

Breadth First Strategy:

In this type of search after the node we proceed to next node and hence depth first.

In this we have to consider the graph with loops.

Indexing Strategy:

There are two types of indexing strategies. One for Dynamic Web Search and

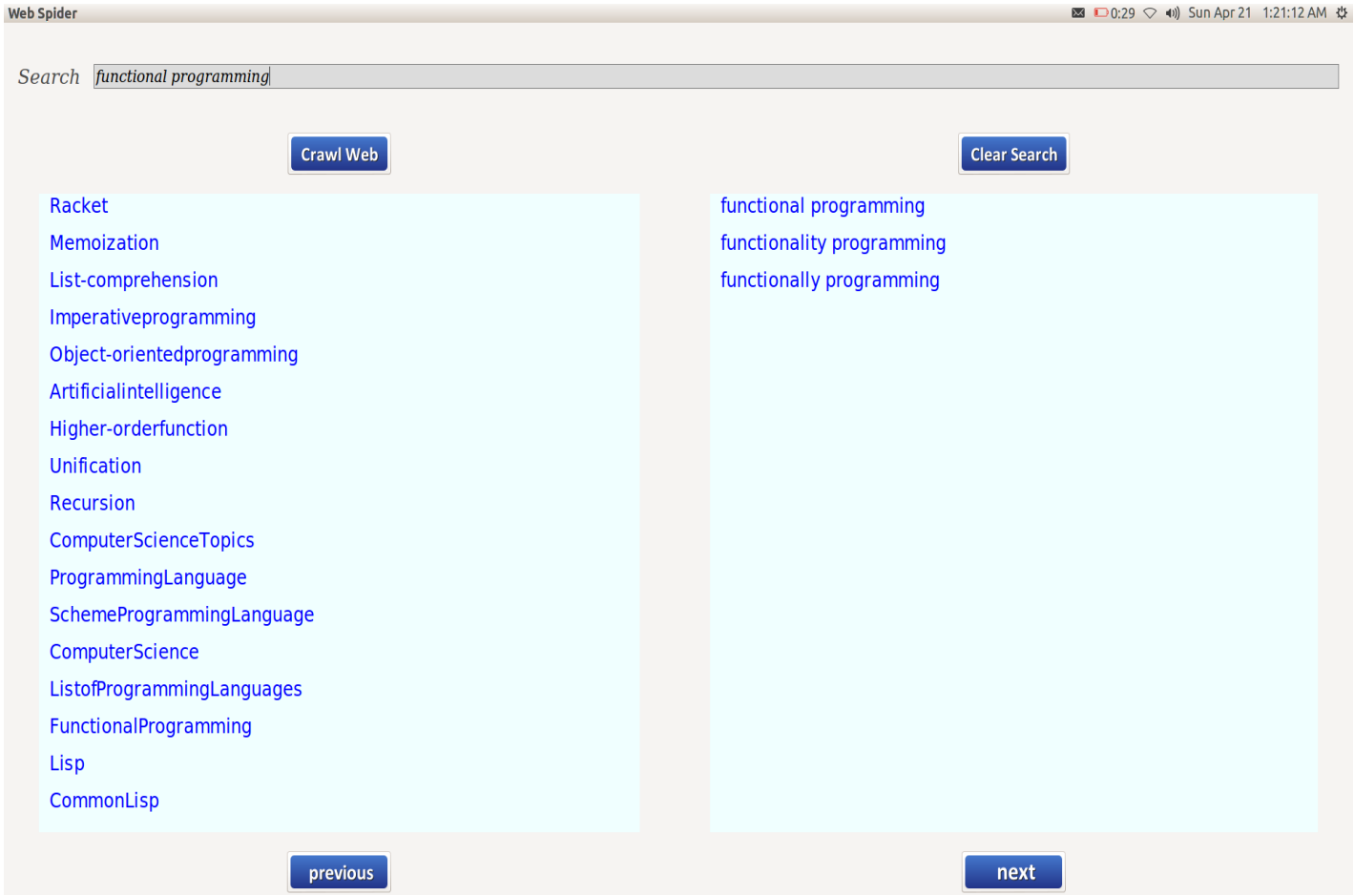
the other for Static Web Search. In the dynamic web search the index is created each and every time the user searches for a result.Whereas in the case of static web search a data base is already constructed and used for the search purpose which is updated form time to time. For large datas and for efficient search static web search is preferred.

The index is modelled as a list of keys (words) where each key consists of a set of values. When provided with the key we can retrieve the set of values. Here the words are modelled as the keys and the pages in which it occurs as the values.

In our case we have considered to store the index in a document and read the docuement every time the program runs. The index can be updated so as to include more data. We read the index and model it so that we finally get a list of all the relevant words along with the pages related to the words. The program looks up into the index and gives the result for a particular search.

*Input and output :*

*for a Query : functional programming*



## *Limitations and Bugs :*

- The words with special characters and words of other languages are not removed which results in storage of some unwanted data in index.
- In the prompt we are not able to auto complete the words or correct the misspelled words.
- There is a limit on the amount of data that can be handled using this process.
- It should be used in linux computers because we used html2text terminal command.
- The prompt will become slow as we type in more words as it should calculate all possible combinations of possible words.