**Multimedia Information Retrieval - Homework 4**

**Name:** Filipe Fernandes Miranda **Student Number:** s1780581

**Answer 1**

The program executes as expected. I used a binary tree to check if an url was already inserted into the queue **(“low-medium difficulty - build a binary tree to check for duplicates”).**

I think my implementation could have been better. In my opinion there was no point in having a queue and a binary tree at the same time since i did not use a linked list the space used was double (queue and binary tree). If I had more time I would have implemented a linked list and then point each node to a pointer of a url. Because I had some problems regarding pointers and spent most of the time fixing them I did not have much more time for anything else.

The binary tree implementation was taken from an answer on Stackoverflow : <http://stackoverflow.com/a/26877608>. A function provided by the assistance to handle directories of websites was used.

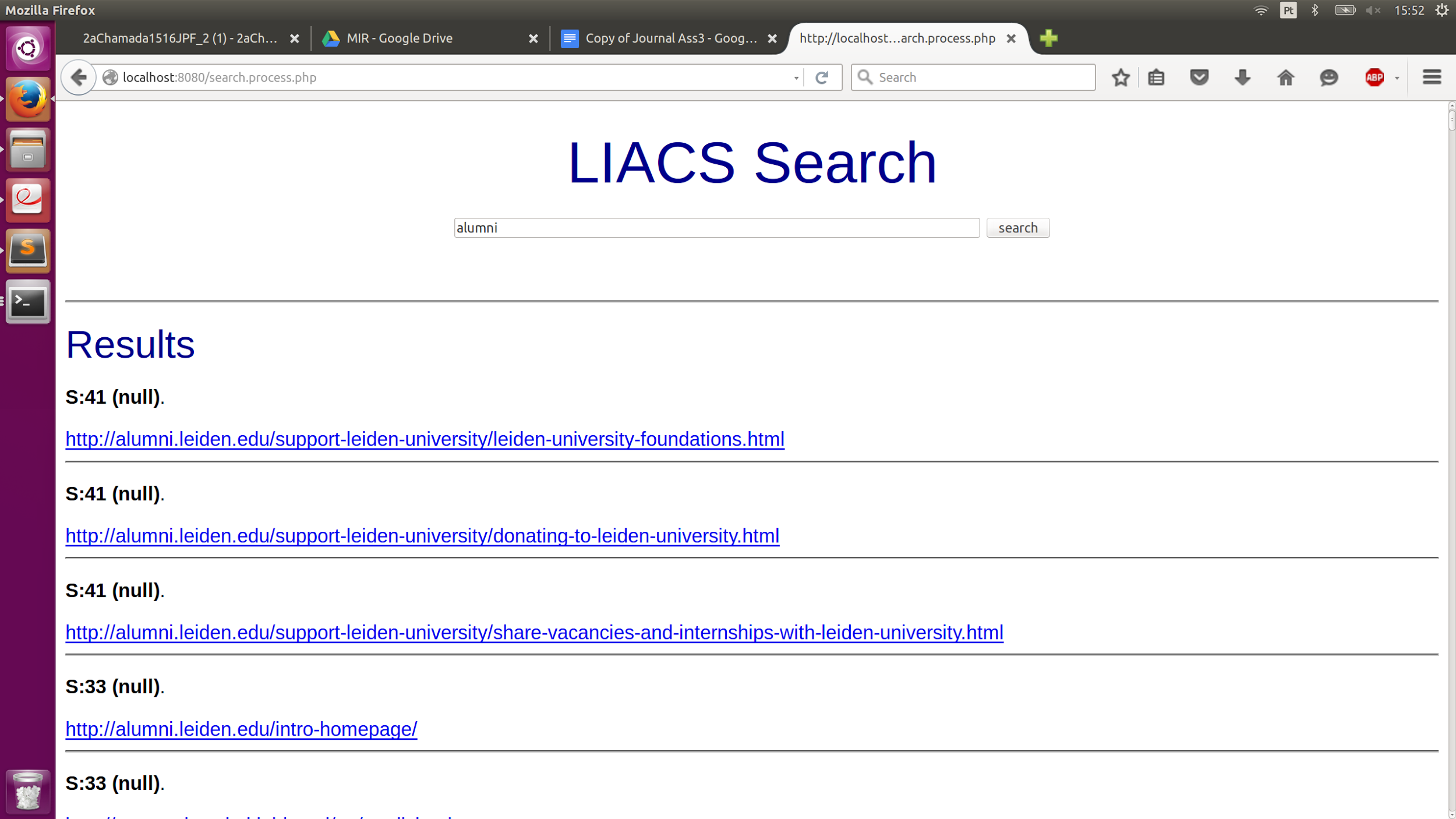
**Answer 2**

I implemented **part of the Medium Difficulty**: Implemented the WebIndex TitleIndex and Linkindex and the Relevance Ranking with those three. Multiple keywords kind of work (you can type two words like “Leiden University” and all the links were each of them appears will show up but is not very well implemented). I also saved the title of each webpage in a file with the url hashed.

The webquery runs the following way after the webspider is executed: it receives a query with one or more words as arguments. For each argument, goes to webindex and title index and opens the file with the argument as the name. For every link read, is put in a binary tree. Each node of a binary tree is composed of a value (url) and score that starts at 1. When inserting by the webindex, the tree creates a node with the url and score=1+4 and when inserting by the titleindex score=1+16. If the link already exists, it just adds the core of the insertion (4 or 16) to the node. In the end, we go through all the links and add the number of links each url has in the linkindex (because each url has a value of one, the number of links inside a file equals the score we must add, pass the tree to a queue ordered by the node’s score. The score is then printed and shown in the webpage.

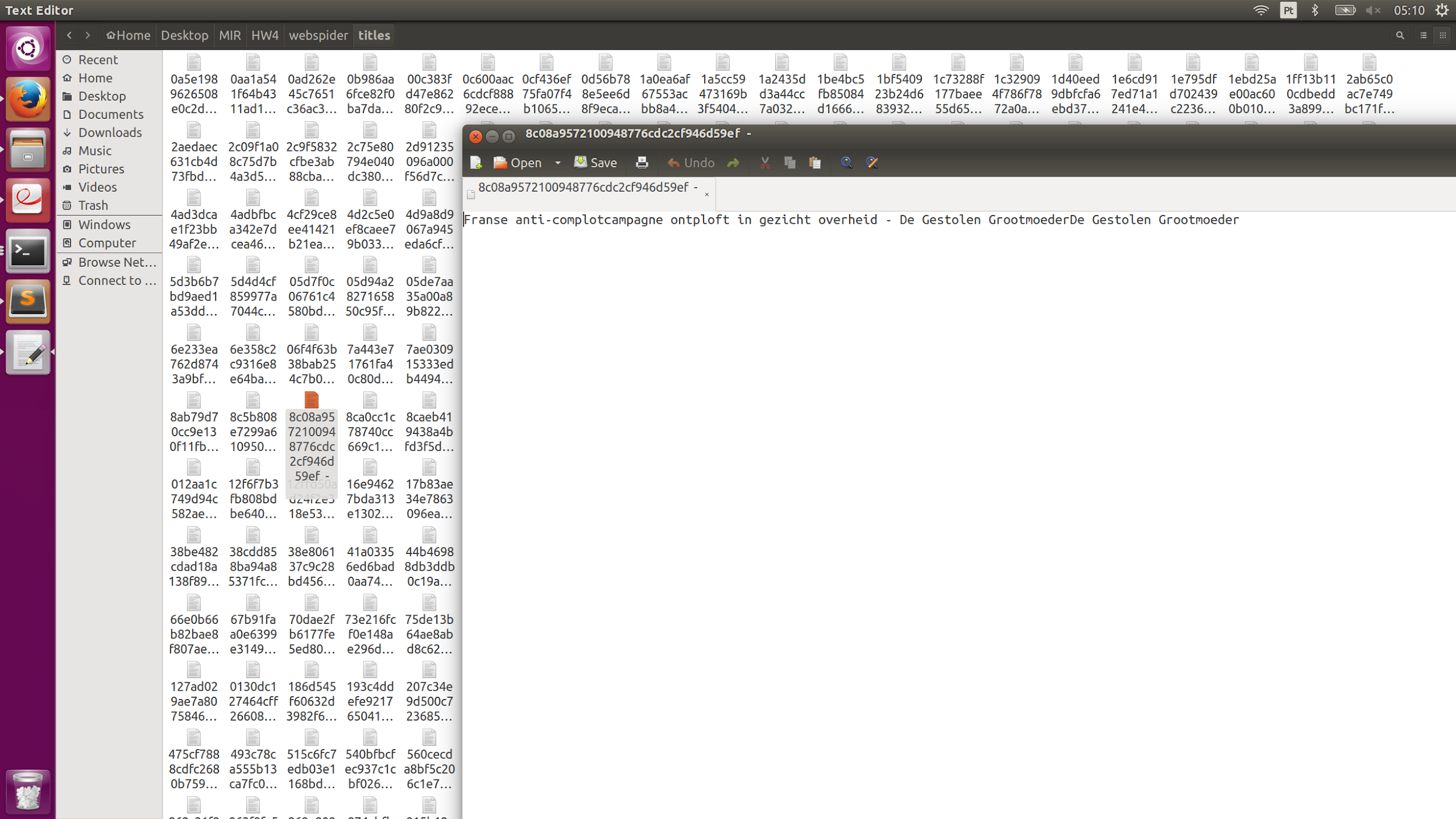
I run the program for 2000 Downloads and in the browser we can see the score, the link and supposedly the title of the webpage. The title is coming null but I could not figure out why, although the files with the titles are being saved:

Example: Search for “alumni” with the results sorted by score:

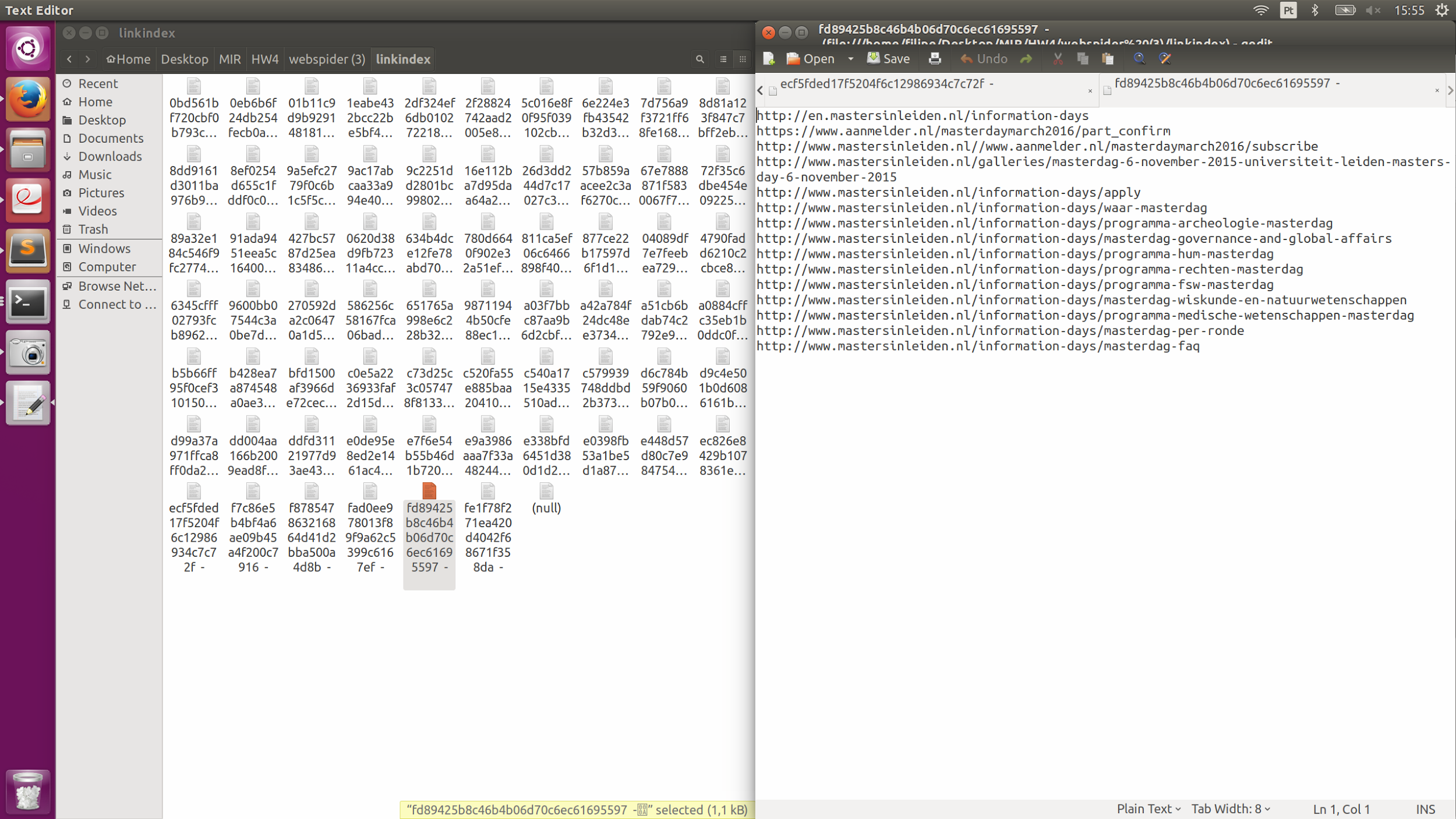


As I stated before, the titles are being saved in files as shown in the picture below, but I could not make it work with the search query I don’t know why.

Image showing the folder with the titles:



Another problem is in saving the linkindex files. I don’t know why but is not saving as many as it should and after a certain time, all the links go to the null file. I could not figure out why. If it was working, the ranking would be preciser:



I think that all the problems I had were because of the hash function I used. I used a call of the system (md5sum) and probably there are collisions occurring or another problem I might be overlooking.