Unix Operating System Project

Nguyen Dang Phuc
1504224

Motivation

Many familiar objects of study can present entirely new aspects if viewed from a different perspective. In programming, one of the most common yet most taken-for-granted practice is the use of brackets, tags and other boundary markers. The construction of numerous structures rely heavily on the use of such boundary markers. Some of the prime examples are programming language source codes, html source code, xml documents. Other, more unfamiliar examples are files formatted to express data structure such as linked lists and trees. These types of data files rely on boundary markers to facilitate clear and feasible data transfer to programs which use their data.

Such heavy use of these boundary markers help machine process these structures easily, yet to human eyes, they are usually difficult to interpret quickly. This project focus on giving this type of format (the use of boundary markers) a visual representation. Visual data often give humans both broad and detailed views of the object studied. This type of another-angle approach or “different” view can be seen in Fourier analysis of signal. Fourier analysis often gives us both a broad and detailed view on the components of a signal. To take it further, the visual data given by Fourier analysis can be incorporated with the sound representation of the signal, enabling us to study the signal both by mathematical understanding and by our inherent senses.

The program written in this project focus mainly on giving web pages a visual representation. By tracking a few entry and exit tags of the html language, the program can produce a graph or a visual **form** of the page’s html code. This translation gives each page a visual identification form (or visual ID). The emphasis on **visual** identification stems from the fact that to humans, html codes of two different pages may appear dissimilar but not cleanly distinct, but shapes and forms on first glance are easily distinguishable. If each pages has a **visual** identity, these forms can extend our view on their obfuscated code structure.

One other reason for the choice of focusing on web pages is the enormous number of them available on the Internet. A common Google search can result in over 1 billion hits and each hit is a different piece of html code with their own “boundary” structure (structure created by the use of entry and exit tags - the focus of this project). This gives the project an unlimited source of test subjects.

How program maps the page?

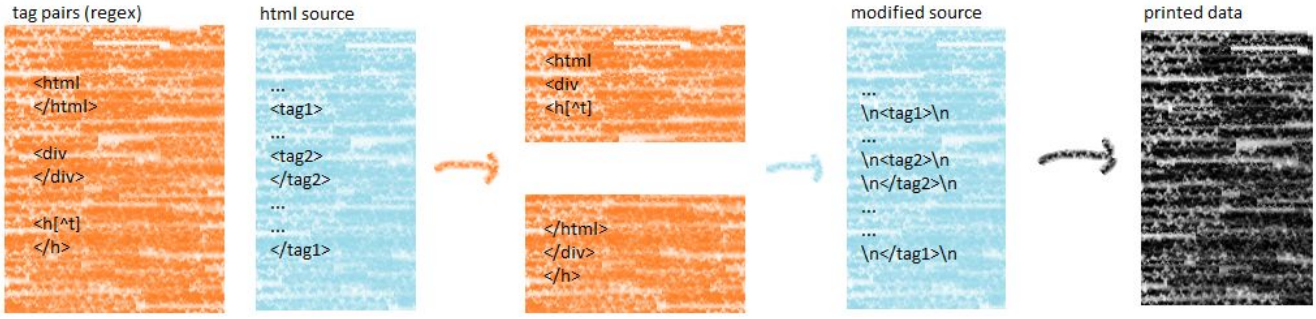


Figure 1

The program takes as input two files. First is a file of pairs. Pairs here refers to the regular expression pairs of entry and exit patterns or tags. Second is the html source file. Then, the program generates two temporary files, one containing all entry tags, the other containing all exit tags. Finally, the program generates a “modified source”, which is a modified version of the html source. This modified version puts every inputted entry and exit tags on their own line (*<tag>* becomes */n<tag>/n*). The reason for this step of preparation is to turn every tag into a *block*, which in Unix is most convenient as a *line.*

The above process can be summarized as the preparation stage, where the outputs are the two lists of tags (separated) and the modified source (convenient for a loop to process).

The program then iterates through each line in the modified source. At every iteration or every line, there is defined a concept called *context.* This *context* can be considered as a packet of four pieces of information:

line = {a tag, not a tag}
direction = {neither, entry, exit}
tier = {a multiple of 10}
line number = 1, 2, ..

line is the content of the line being read in modified source. The focus is on whether or not it is a tag. If it is not a tag, it is mostly ignored in the mapping task.

direction refers to the direction of the tag in the *context.* If the *line* is not a tag, *direction* is neither, otherwise, it is either entry or exit.

tier refers to the level of “depth” the *context* is at. Every entry tag encountered raises the *tier* value by 10 and every exit tag decreases the *tier* value by 10. This *tier* value is the key

component in mapping the modified source to a visual form. Every time the *context* **enters** a tag, it prints that tag at the end of a space whose size is proportional to the *tier* value, the similar thing is applied when *context* **exits** a tag. This can be understood from Figure 2 (below).

line number is the number of the line being read. Note that this line number is of the modified source, not the html source.

This is an example output of the *tier* printing scheme (with line number attached)

Figure 2

This output is produced from two input files: tags (content shown below) and the content of the url https://en.wikipedia.org/wiki/Unity_(game_engine)

```
<script
</script>

<tr
</tr>

<td
</td>

<body
</body>

<html
</html>

<p
</p>

<ul
</ul>

<div
</div>

<h[^t]
</h>
```

The tags used to map this page are shown on the left. Note that these are regular expressions and they do offer flexibilities as well as limitations.

Visuals

The example output shown in Figure 2 is one of two main output forms of the program. The second form of output is a list of *tier* value at every tag encounter (the key component to mapping the page.) Here is an example.

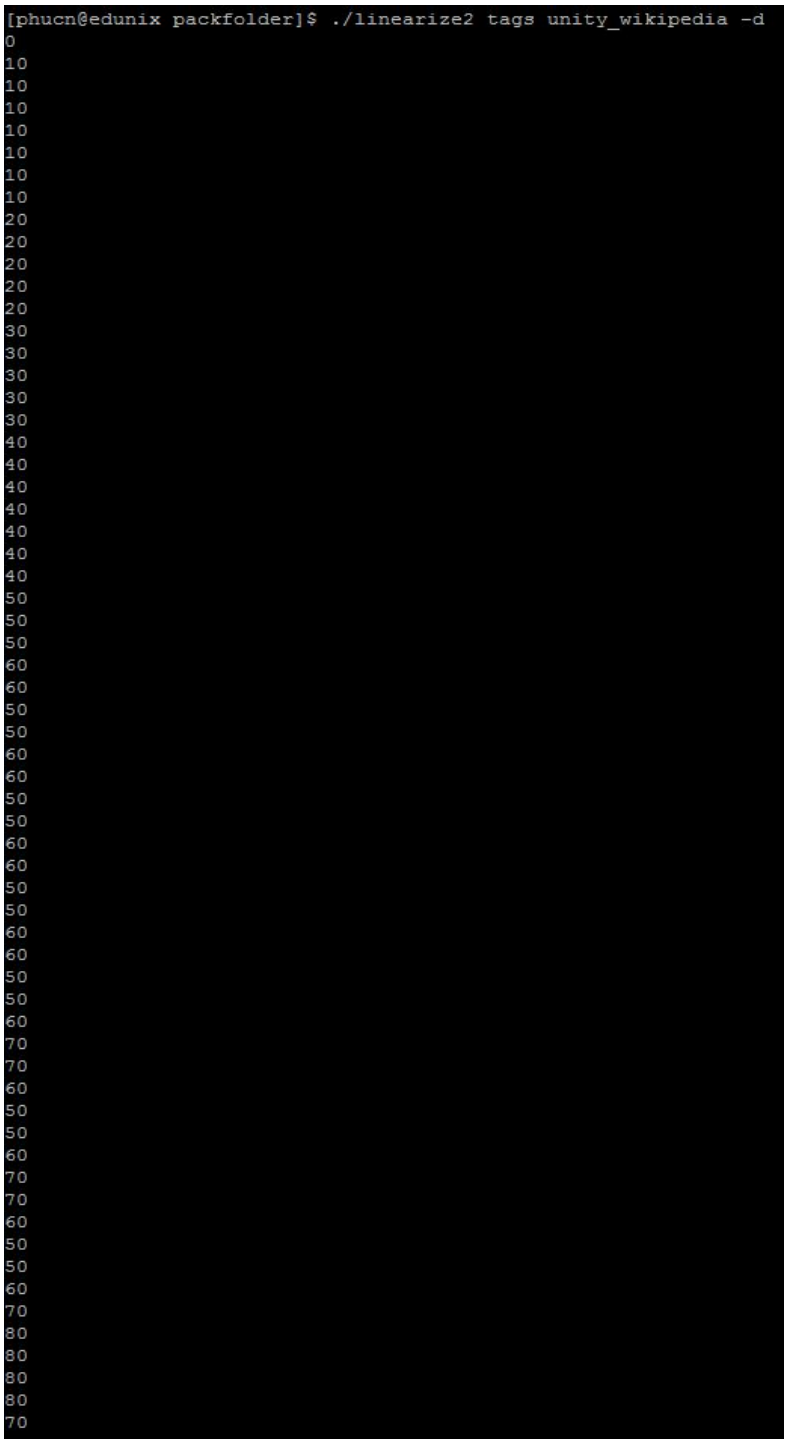


Figure 3

This does not look very informative as is. But after graphing the values from the output …

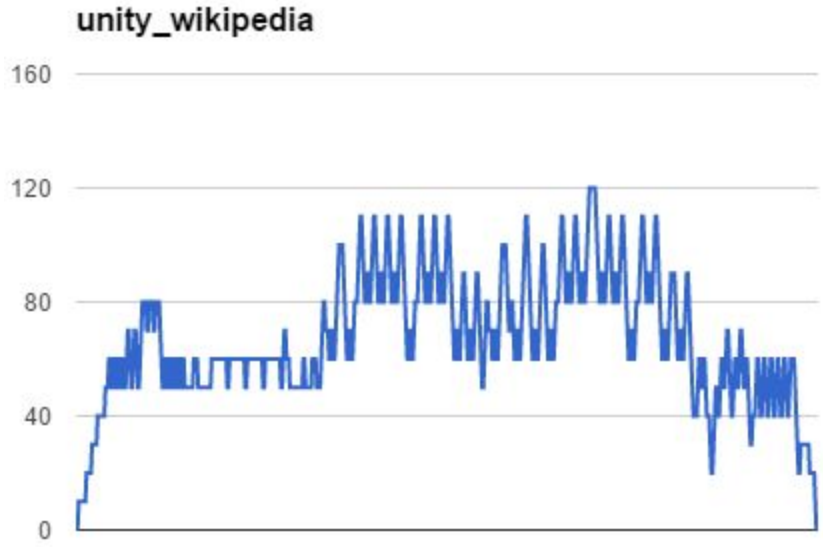


Figure 4 - Visual form of Wikipedia page on Unity (game engine)

Apply the same to 3 other web pages and view the results together

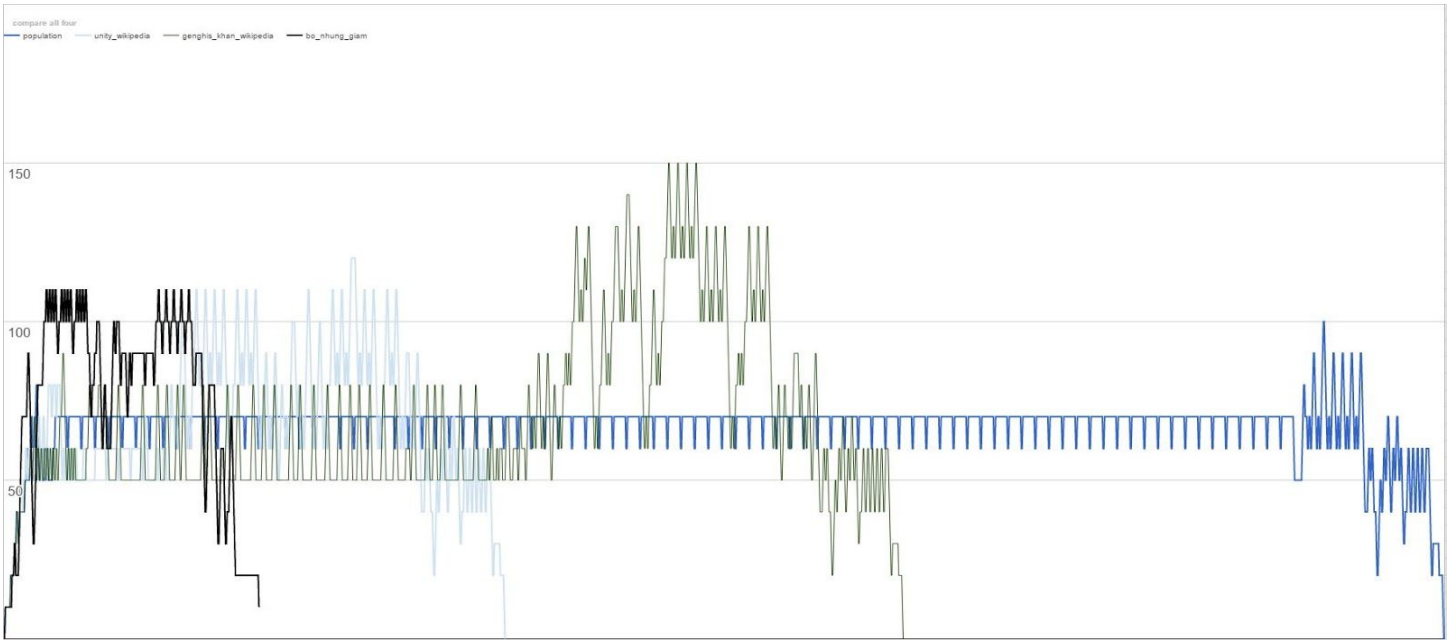


Figure 5 - Four web pages' graphs viewed together
(horizontal length represents line count of modified source)

In order from shortest to longest (horizontally)
http://monngonmoingay.com/bo-nhung-giam/
https://en.wikipedia.org/wiki/Unity_(game_engine)
https://en.wikipedia.org/wiki/Genghis_Khan
https://en.wikipedia.org/wiki/List_of_cities_proper_by_population

Figure 5 shows the different graphs of the four pages. But while the distinction is certainly obvious in Figure 5, Figure 6 re-graphs the pages to further emphasize their different forms.

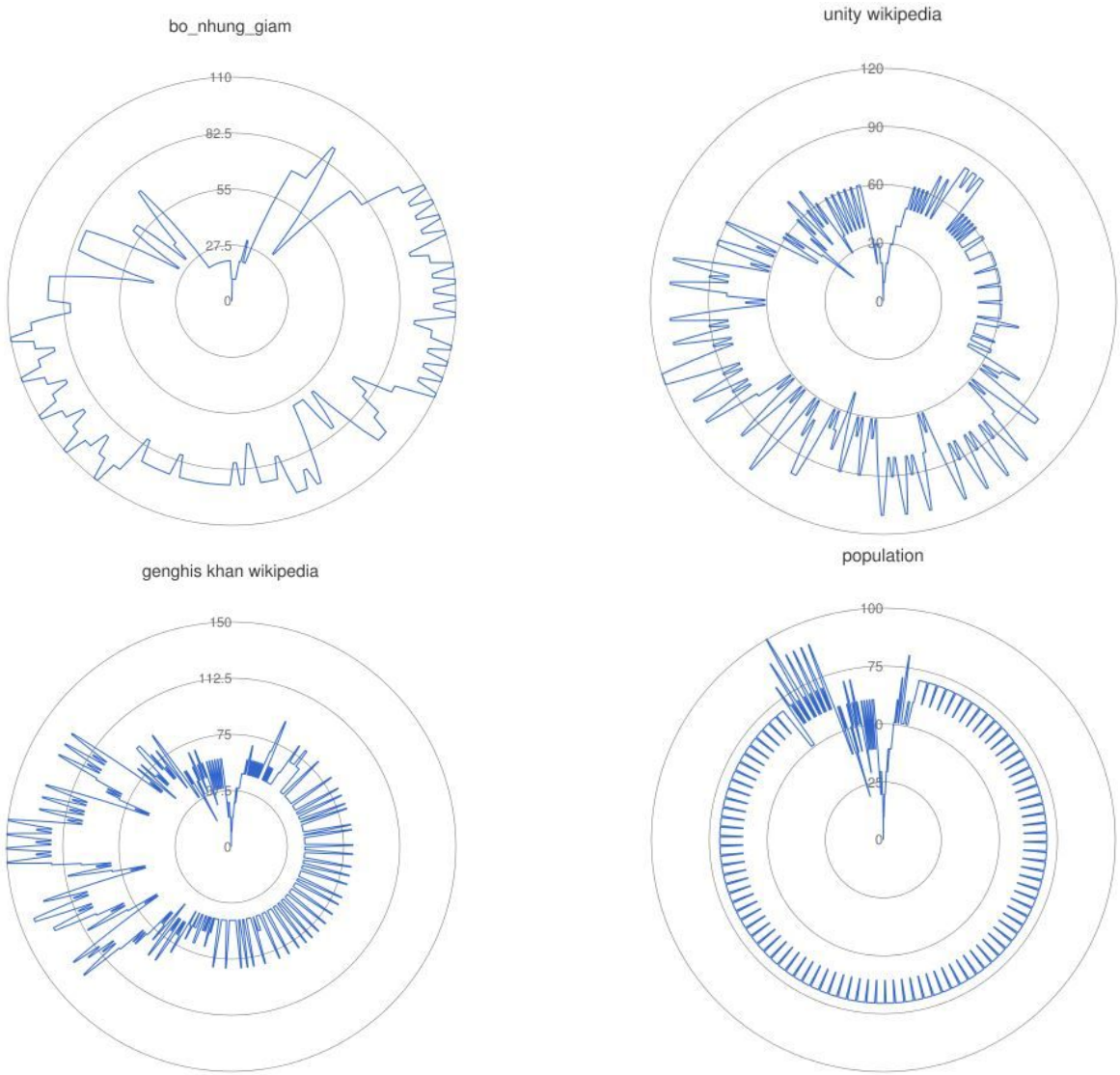


Figure 6 - Polar graphs

It can be seen that genghis_khan_wikipedia has the deepest level of nesting.

A remark about figure 5: Observe that in figure 5, the black graph's right leg does not touch the ground. This can be caused either unbalanced tags or limitations in the regular expression form of the tags inputted.

Remark on the form of the graph:
Every rise indicates a sequence of entry tag encounters.
Every drop indicates a sequence of exit tag encounters.
The flat sections indicate a sequence of alternating entries and exits of matching tags balancing rise and drop continually, thereby creating a flat line.

The program guide

The script name: linearize (naming choice is purely of personal taste and has no related reasons)
Helper scripts: pack, sedcmd
Share resource: wd (contains operating directory - the directory which contains all of these files)

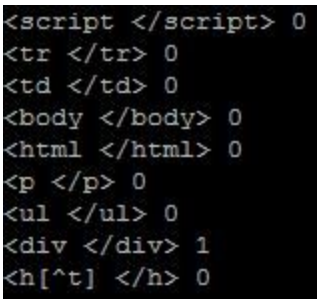
Table lists some possible commands

Command	Does
./linearize --help	show this table's content
./linearize <tag pair file> <html source>	print "text graph", balance table
./linearize <tpf> <s> blank --print-modified-source	print modified source, "text graph", balance table
./linearize <tpf> <s> -n	same as "./linearize <tpf> <s>" with line number attached in "text graph"

./linearize <tpf> <s> -n -pms	same as "./linearize <tpf> <s> blank --print-modified-source" with line number attached in "text graph"
./linearize <tpf> <s> -d	print *tier* values and balance table
./linearize <tpf> <s> any any -nb	not print balance table

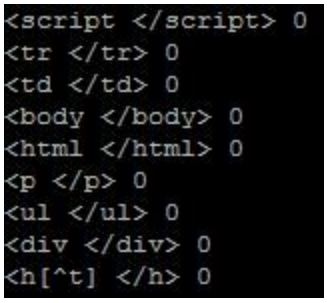
Balance table

As remarked about unbalanced graph in Figure 5, here is the balance table for that graph (black graph in Figure 5)



0 : number of entries = number of exits
1 : number of entry <div is 1 more that number of exit

balance table of bo_nhung_giam



All other three graphs are balanced, they balance table look the same for the same set of tags.

balance table of unity_wikipedia, genghis_khan_wikipedia, population