Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Sistemas Operativos 2

Practica 2 – Manual Tecnico

Integrantes GRUPO 15

201313889	Hilbert Josué Perucho
201408489	Paul Steve Contreras
201801262	Juan Jose Ramos
201807266	Luis Fernando Velasquez

1 de abr. de 22

Herramientas necesarias para realizar la practica

- Sistema Linux
- IDE para realizar el código en este caso se realizará en visual code
- Almacenamiento de disco disponible
- Versión más reciente de GO

Librerias implementadas

```
import (
        "bufio"
        "crypto/sha1"
        "encoding/hex"
        "fmt"
        "os"
        "strconv"
        "strings"

        "github.com/euskadi31/go-tokenizer"
        "github.com/gocolly/colly"
)
```

Librería go-tokenizer: es una herramienta que permite realizar un parseo en la cadena recibida esto con el fin de poder separar la cadena conforme a la necesidad que se desee.

Ejemplo de estructura.

```
for {
    tt := z.Next()
    if tt == html.ErrorToken {
        // ...
        return ...
    }
    // Process the current token.
}
```

Librería colly: es un marco de trabajo de rastreo web implementado en go. Puede iniciar mas de un 1k de solicitudes por segundo en un solo núcleo; proporciona un conjunto de interfaces en forma de funciones de devolución de llamada, que pueden implementar cualquier tipo de rastreador; confiando en la biblioteca de goquery puede seleccionar elementos web como jquery.

La función getReader es la encargada de analizar el texto que se vaya recibiendo y su separado entre cada fila será cuando identifique un salto de línea.

```go
func getReader(text string) string {
    reader := bufio.NewReader(os.Stdin)
    fmt.Print(text)
    t, _ := reader.ReadString('\n')
    t = strings.Replace(t, "\n", "", -1)
    return t
}
```

La siguiente función es la encargada de leer la COLA hasta cuando la posición sea Nr – 1 “colaGeneral” es un arreglo del struct encargado de almacenar la informacion.

```go
colaGeneral = llenarCola(intCola, intNr, url, colaGeneral)
//fmt.Println(hijos)
done := make(chan struct{})
no_mono := 1
for i := 0; i <= intCola; i = i + (intCola / intMonos) + 1 {
    paso := i + (intCola / intMonos)
    if paso > intCola {
        paso = intCola
    }
    go buscarMono(i, paso, f, done, colaGeneral, no_mono)
    no_mono++
    //scraper("0", intCola, intNr, url, intMonos)
}
```

La función buscarMono se llama concurrentemente con cierto rango de la cola, y escribe un archivo al terminar. Al haber finalizado retornar un CHANNEL para indicar el hilo en el que se terminó de ejecutar.

```go
func buscarMono(inicio int, fin int, f *os.File, done chan struct{}, colaGeneral []Mono, no_mono int) {
    for i := inicio; i < fin; i++ {
        monoActual := colaGeneral[i]
        monoActual.mono = no_mono
        _, err := f.WriteString(fmt.Sprint(monoActual))
        if err != nil {
            panic(err)
        }
    }
    done <- struct{}{}
}
```

Función SCRAPER, es la encargada de realizar la extracción de información del sitio web. Se extrae la información en código HTML y con ello los datos que estén almacenados en la base de datos.

```
func scraper(origen string, tam_cola int, url string, mono_id int, colaGeneral []Mono) (links []string,
    base_url := "https://es.wikipedia.org"

    if !strings.Contains(url, base_url) {
        url = base_url + url
    }

    c := colly.NewCollector(
        colly.AllowedDomains("es.wikipedia.org", "en.wikipedia.org"),
    )
    // CONTAR LA CANTIDAD DE PALABRAS

    tokens := 0
    cantLinks := 0
```

La función hijos, es la encarga de reconocer los hijos de cada uno de los hilos que se analicen en el método scraper.

```
func hijos(intCola int, nivel int, nivelActual int, origen string, link string, mono_id int, colaGenera
    //fmt.Println(nivelActual)
    if nivelActual >= nivel {
        //fmt.Println("saliendo")
        return colaGeneral
    } else {
        links := []string{}
        links, cola = scraper(origen, intCola, link, mono_id, colaGeneral)
        nivelActual++
        if len(cola) < intCola {
            for _, element := range links {
                cola = hijos(intCola, nivel, nivelActual, getSha256(link), element, mono_id, cola)
                if len(cola) >= intCola {
                    break
                }
            }
            return cola
        } else {
            return cola
        }

    }
    return cola
}
```

En función de MAIN se solicitan cada uno de los datos para almacenar la información. De igual manera se almacenan en la cola para ser escrita en un archivo.

```go
func main() {
    colaGeneral := []Mono{}
    monos := getReader("Cantidad de monos buscadores: ")
    intMonos, _ := strconv.Atoi(monos)

    cola := getReader("Tamaño de la cola de espera: ")
    intCola, _ := strconv.Atoi(cola)

    nr := getReader("Nr: ")
    intNr, _ := strconv.Atoi(nr)

    url := getReader("URL: ")

    file := getReader("Nombre de archivo: ")

    //fmt.Println(file)
    f, err := os.Create(file + ".txt")
    if err != nil {
        panic(err)
    }
    colaGeneral = llenarCola(intCola, intNr, url, colaGeneral)
```

Ejecución del programa.

```
hperucho@ubuntu:~/Desktop/SO2_1S2022_P2_G15$ go run main.go
Cantidad de monos buscadores: 5
Tamaño de la cola de espera: 5
Nr: 4
URL: https://es.wikipedia.org/wiki/Cristiano_Ronaldo
Nombre de archivo: cr7
{0 0 0 0926ccb697739db66808aca54e1cc763ad062a45 https://es.wikipedia.org/wiki/Cristiano_Ronaldo 0}
{0926ccb697739db66808aca54e1cc763ad062a45 0 0 53d32fd5d9af34faf4470c5425dd45142782e7df https://es.wikipedia.org/wiki/Alfabeto_Fon%C3%A9tico_Internacional 0}
{b9985e88a268d931f499bf704180d6e3631d2283 0 0 f14bed3569ff2b020b24a5d5a36d5bf5f3540829 https://es.wikipedia.org/wiki/Idioma_espa%C3%B1ol 0}
{237afa51ec7f62e9e8e5b035188a25007de85c65 0 0 8c9ea90516f5a20b12b9bd2735d3b04234d4488e https://es.wikipedia.org/wiki/Lenguas_indoeuropeas 0}
{3f529e8e930256e4e039fd63d9ebe22ca5d9e6e5 0 0 23dcded149ea6e87f9d79512175d9aff5cf27fca https://es.wikipedia.org/wiki/Familia_de_lenguas 0}
```