

LazyWGET: Documentație

Parcursarea de tip Breadth-First Search a unui arbore de linkuri

RUSLAN GAITUR

CRISTIAN BUDALA

2025

Cuprins

1. Prezentare Generală	3
2. Instalare	3
2.1 Dependințe	3
2.2 Configurare	3
3. Utilizare	3
3.1 Executarea Scriptului	3
3.1.1 Prima Rulare (L_0)	3
3.1.2 Rulări Ulterioare ($L_{N>0}$)	4
3.2 Resetarea Scriptului	4
3.3 Interfața Vizuală	4
3.3.1 Modul Inițial (<code>./lwget <URL></code>)	4
3.3.2 Modul Recursiv (<code>./lwget</code>)	4
3.3.3 Diagrama Fluxului de Afișare	4
4. Structura Ierarhiei de Directoare	5
4.1 Structura Superioară a Ierarhiei	5
4.2 Diagrama Ierarhiei	5
4.3 Consecințe ale Designului	5

5. Arhitectura Proiectului	6
5.1 Promisiune	6
5.2 Execuție	6
5.3 Limitări	6
6. Depanare	7

1 Prezentare Generală

LazyWGET este un script Bash care realizează o parcurgere de tip Breadth-First Search (BFS) a unui arbore cu rădăcina în primul URL introdus de utilizator, iar nodurile reprezintă referințe (HREF-uri) de la nivelul anterior. Acesta descarcă referințele (promisiuni) câte un “nivel de adâncime” pe rând, oprind execuția după fiecare nivel.

Scriptul este proiectat să fie:

- **Secvențial:** Nu execută arborele complet de recursivitate automat. Așteaptă apelarea de către utilizator pentru a trece la adâncimea $N + 1$.
- **Programatic:** Păstrează starea cozii de procesare sub forma unei baze de date de URL-uri (promisiuni).
- **Eficient:** Previne buclele circulare verificând istoricul înainte de a adăuga noi URL-uri în coadă.

2 Instalare

2.1 Dependințe

Asigurați-vă că următoarele utilități standard Linux sunt instalate în mediul dumneavoastră:

- bash (Mediul Shell)
- wget (Descărcare rețea)
- grep, awk, cut (Pipeline-uri de procesare text)

2.2 Configurare

1. Salvați scriptul ca `lwget`.
2. Acordați permisiuni de execuție:

```
1 chmod +x lwget
```

3 Utilizare

3.1 Executarea Scriptului

3.1.1 Prima Rulare (L_0)

Pentru a inițializa proiectul și a descărca pagina rădăcină, introduceți URL-ul root ca argument.

```
1 ./lwget <URL>
```

Funcție: Nivelul este setat la 0. Fișierul rădăcină este descărcat și parsat.

3.1.2 Rulări Ulterioare ($L_{N>0}$)

Pentru a avansa parcurgerea la următoarea adâncime, rulați scriptul fără argumente.

```
1 ./lwget
```

Funcție: Procesează toate promisiunile de nivel N , generează promisiunile de nivel $N + 1$ și incrementează contorul adâncimii.

3.2 Resetarea Scriptului

Pentru a șterge memoria cache, a elimina fișierele descărcate și a reseta contorul de nivel la 0:

```
1 ./lwget -r
2 ./lwget --reset
```

Funcție: Elimină recursiv `lwget_cache`.

3.3 Interfața Vizuală

Interfața vizuală a scriptului este definită de textul returnat la *Standard Output*. Interacțiunea grafică este dinamică și depinde strict de argumentele furnizate la lansarea unei comenzi din script.

3.3.1 Modul Inițial (`./lwget <URL>`)

La prima rulare, interfața afișează procesarea unei singure entități (root URL), însoțită de un mesaj de confirmare al finalizării.

3.3.2 Modul Recursiv (`./lwget`)

La rulările ulterioare, interfața vizuală devine o listă secvențială de operațiuni. Deoarece scriptul procesează coada de promisiuni, utilizatorul vede un flux continuu de linkuri, reprezentând promisiunea de la nivelul (L_N) parsată la momentul respectiv.

3.3.3 Diagrama Fluxului de Afișare

[INPUT UTILIZATOR]	[RĂSPUNS VIZUAL]
	+---- ./lwget	-->	---	processing level N	----
				"style.css"	
				"script.js"	
				"image.png"	
				.	
				.	
				.	
			---	finished processing level N	----

```
|
+--- (Eroare)          --> "You must enter an URL!"
                        --> "You must enter a single URL!"
```

4 Structura Ierarhiei de Directoare

Spre deosebire de uneltele standard care doar afișează text la *Standard Output*, LazyWGET construiește o structură sub formă de ierarhie a fișierelor generate, permițând utilizatorului să interacționeze cu lucrul făcut de script.

4.1 Structura Superioară a Ierarhiei

Interfața principală prin care utilizatorul interacționează cu rezultatele este structura directoarelor din cache. Aceasta funcționează ca un “dashboard” static al proiectului:

- **Zona de Control (.metadata):** Acționează ca panoul de stare al aplicației. Fișierele precum `current_level.txt` oferă o indicație imediată a progresului curent în arborele BFS, servind la stocarea și utilizarea ulterioară a datelor de către script.
- **Zona de Conținut (downloads):** Aici structura site-ului root este reconstruită fizic, permițând utilizatorului să inspecteze resursele (HTML, CSS, Imagini, ...) în formatul lor nativ.

4.2 Diagrama Ierarhiei

```
lwget_cache/
|-- .metadata/          # Urmărirea stării interne
    |-- current_level.txt # Stochează nivelul curent care trebuie parsat
    +-- promises.txt      # Baza de date a promisiunilor sortate după nivel
|-- downloads/          # Stocarea conținutului
    |-- downloaded.html   # Cel mai recent fișier HTML procesat
    +-- assets/           # Resursele până la adâncimea nivelului curent
```

4.3 Consecințe ale Designului

Structura directoarelor implementează principii specifice pentru a susține natura “Lazy” a scriptului:

1. **Feedback Secvențial:** Deoarece execuția este secvențială, ierarhia de directoare oferă pauze vizuale naturale în construirea sa. Utilizatorul poate verifica integritatea fișierelor din `downloads` înainte de a lansa comanda pentru nivelul următor ($N + 1$), prin intermediul comenzii `ls`.
2. **Separarea Preocupărilor:** Designul ierarhiei separă strict logica de navigare (metadatale) de conținutul efectiv, prevenind aglomerarea vizuală a directorului de lucru.

5 Arhitectura Proiectului

Scriptul se bazează pe o coadă de execuție sub formă de promisiuni pentru a gestiona graful de parcurgere.

5.1 Promisiune

Mecanismul de bază al parcurgerii arborelui este promisiunea. O promisiune este un URL identificat în nivelul curent (L) care este programat pentru a fi vizitat în nivelul următor ($L + 1$). Promisiunile sunt stocate într-un fișier text de perechi (`promises.txt`) în formatul:

LEVEL | URL

5.2 Execuție

1. **Inițializare (L_0):** Utilizatorul furnizează un URL de pornire (seed sau root). Scriptul descarcă rădăcina (root), parsează HTML-ul pentru hyperlink-uri (*href*) și salvează aceste linkuri ca promisiuni pentru Nivelul 1.
2. **Execuție Incrementală (L_N):** La rulările ulterioare, scriptul citește fișierul de metadate pentru a determina nivelul curent. Recuperează toate promisiunile etichetate cu nivelul curent, descarcă resursele, extrage noile link-uri și le salvează ca promisiuni pentru Nivelul $N + 1$.

5.3 Limitări

- **Detectarea URL-urilor:** Scriptul utilizează `grep` pentru a găsi link-urile referite de fișierele HTML. Acestea sunt puse direct în coada bazei de date de promisiuni. Suportă strict `http` și `https`.
- **Detectarea Asset-urilor:** Scriptul folosește Regex (`grep`) pentru a identifica extensiile fișierelor. Extensiile cunoscute (Tabelul 1) sunt tratate ca noduri terminale, deoarece sunt descărcate dar nu sunt scanate pentru potențiale promisiuni.
- **Gestionarea Duplicatelor:** Scriptul utilizează `grep -Fq` pentru a scana baza de date de promisiuni înainte de a o actualiza pentru nivelul următor, asigurând unicitatea promisiunilor.

Categorie	Extensii
Imagini	jpg, jpeg, png, gif, webp, bmp, ico, tiff, svg
Documente	pdf, doc, docx, xls, xlsx, ppt, pptx, txt, csv
Cod & Date	css, js, json, xml, map
Media	mp3, mp4, wav, webm, m4a
Arhive	zip

Tabel 1: Clasificarea extensiilor de asset-uri terminale.

6 Depanare

Problemă	Cauză Posibilă	Soluție
“You must enter an URL!”	Scriptul a rulat fără argumente la Nivelul 0.	Furnizați rădăcina: <code>./lwget <URL></code>
“You must enter a single URL!”	Mai mult de 1 URL rădăcină furnizat la Nivelul 0.	Rulați <code>./lwget</code> cu un singur URL rădăcină.

Tabel 2: Erori comune de execuție și soluții.