UNIVERSITA' DI CAMERINO
CORSO DI LAUREA IN INFORMATICA

RELAZIONE PROGETTO SISTEMI OPERATIVI

Federico Casenove, Mat. 099210
Daniele Serafini, Mat. 097845

Data consegna: *28 Settembre 2018*

INTRODUZIONE

La seguente relazione riguarda il progetto realizzato come esame della parte di Laboratorio di Sistemi Operativi. Come da consegna, il programma, dati in input da linea di comando file e/o directory genererà un file di output (di default swordx.out) contenente la lista delle parole lette nei file passati in input con le relative occorrenze. In oltre è possibile specificare dei parametri per approfondire la ricerca.

UTILIZZO

Invocazione: ***./swordx [options] [inputs]***

Dove ***[options]*** *sono le diverse opzioni che il programma mette a disposizione. Mentre* ***[inputs]*** *sono i file o directory che il programma andrà poi ad elaborare.*

PARAMETRI

Al programma è possibile passare le seguenti opzioni:

- ***help(-h):***
    - *Stampa a video delle istruzioni con printf();*
- ***recursive(-r):***
    - *Controlla se all'interno delle directory sono presenti delle sottodirectory, il controllo è situato nel metodo;*
- ***follow(-f):***
    - *Controlla se il file o cartella è un symboliclink, se lo è, reperisce il path originale del file e lo esegue;*
- ***exclude(-e):***
    - *Legge il file preso da linea di comando e lo esclude dall'analisi delle occorrenze;*
- ***alpha(-a):***
    - *Nel conteggio delle occorrenze vengono elaborate solo le parole unicamente composta da caratteri alfabetici;*
- ***min(-m):***
    - *Nel conteggio delle occorrenze vengono elaborate solo le parole composte da un minimo numero di caratteri;*
- ***ignore(-i):***
    - *L'elenco delle parole da ignorare durante il conteggio delle occorrenze;*
- ***sort by occurency(-s):***
    - *Ordina le parole in base alle occorrenze.*

SORGENTI

Il progetto è suddiviso in cinque file sorgente:

- ***swordx.c***
    - *Si occupa del parse da linea di comando e di tutte le operazioni riguardanti la scansione dei file e delle directory;*
- ***args.c***

- *Struttura contenente gli argomenti presi in input;*

- ***trie.c***
  - *Un trie che gestisce il salvataggio delle parole lette dai file;*

- ***bintree.c***
  - *Un binary tree necessario per l'inserimento in modo ordinato delle parole lette;*

- ***linkedstack.c***
  - *Una linked list necessaria a explude per il salvataggio dei file da escludere;*

- ***log.c***
  - *Una linked list necessaria al salvataggio delle informazioni del logger.*

LIBRERIE

Tutte le librerie utilizzate dal programma fanno parte della standard library di C (glibc).

*Le librerie **POSIX** utilizzate sono dirent.h (per analizzare file e cartelle) e sys/stat.h (per distinguere i tipi di file/directory/symlink).*

IMPLEMENTAZIONE

*Come struttura dati per il salvataggio delle parole e' stata scelto il trie, che contiene un contatore che viene riempito soltanto quando una parola e' stata totalmente inserita e un puntatore ai figli possibili limitati dal **CHARSET** (che in questo caso sono 26 piu' 10 numeri), il vantaggio di questa struttura e' nell'inserimento delle parole, da notare comunque lo spreco di spazio durante la creazione dei nodi figli, che vengono impostati a **NULL** durante l'inserimento delle parole.*
E' stato inoltre implementato un binary tree per l'ordinamento in base alle occorrenze. Quando l'opzione "sort by occurrencies" e' impostata, viene effettuato inizialmente l'inserimento nel trie, poi la lettura delle parole che vengono inserite all'interno del binary tree ordinatamente in base alle occorrenze.
Per l'opzione exclude e' stata implementata una linked list, che viene riempita dai file contenuti dal file passato in input.
Gli argomenti presi da console vengono salvati nella struttura args.
Per il log e' stata implementata un semplice linked list, che viene riempita durante la lettura del file.

Per l'impostazione delle flag che non richiedono argomento sono state settate le seguenti macro:

define recursive_flag (1<<0)
define follow_flag (1<<1)
define alpha_flag (1<<2)

define sbo_flag (1<<3)

e viene impostata inizialmente una byte flag. Per il controllo dell'opzione viene chiamata un'altra macro che effettua l'AND tra le due, restituendo un valore diverso da zero se la flag e' impostata:

ISFLAGSET*(bitoption, flag) (bitoption & flag)*

TESTING

*Per i test sono stati utilizzati **Valgrind** e **GDB**. Valgrind e' stato utile per correggere gli errori di memoria non liberata, essendo molto verboso sono stati anche riscontrati "problemi" per quanto riguarda l'inizializzazione di alcune variabili. GDB e' stato utilizzato come debugger, molto comodo per leggere passo passo il programma in runtime e vedere come lavorano le funzioni ed il flusso dei dati all'interno del programma.*