

Laboratorio 6 — Analizzatore di Log da Riga di Comando

Assignment — logcli.c

Università di Catania — Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

Programmazione 1 e Laboratorio (Canale O-Z)

Descrizione del Programma

Si scriva un programma in linguaggio C, chiamato `logcli.c`, che analizzi un file di log testuale ed elabori un report riassuntivo salvandolo su un secondo file di testo.

Il file di input (ad esempio, `input.log`) contiene una sequenza di eventi registrati nel seguente formato testuale:

```
LEVEL;TIMESTAMP;MESSAGE
```

I livelli di log ammessi nel primo campo sono esclusivamente:

- INFO
- WARN
- ERROR

Il programma deve essere eseguito da riga di comando passando come parametri i nomi dei file di input e output nel modo seguente:

```
./logcli input.log report.txt
```

Requisiti di Implementazione

Il programma deve essere strutturato nei seguenti punti:

PUNTO A: Argomenti da riga di comando: Verificare che siano passati esattamente due argomenti oltre all'eseguibile: il nome del file di input e il nome del file di output. In caso contrario, stampare su `stdout`:

```
Errore: uso corretto ./logcli input.log report.txt
```

e terminare con `exit status 1`.

PUNTO B: Gestione Apertura File: Aprire il file di input in lettura testuale ("r") e il file di output in scrittura testuale ("w"). Se l'apertura del file di input fallisce, stampare:

`Errore: impossibile aprire il file di input.`

e uscire con exit status 1. Se fallisce l'apertura dell'output, stampare:

`Errore: impossibile aprire il file di output.`

e uscire con exit status 1.

PUNTO C: Lettura e Parsing dei Log: Leggere il file riga per riga (utilizzando `fgets` su un buffer di dimensione massima di 512 byte). Rimuovere il newline finale '\n' e dividere la riga nei tre campi (LEVEL, TIMESTAMP, MESSAGE) usando `strtok` con separatore ';'.

Una riga è considerata valida se e solo se:

- contiene esattamente tre campi separati da ';' ;
- il primo campo è uno dei tre livelli validi (INFO, WARN, ERROR);
- il timestamp e il messaggio non sono vuoti.

Se una riga è malformata, il programma non deve interrompersi ma incrementare un apposito contatore delle righe non valide.

PUNTO D: Analisi Statistiche: Calcolare le seguenti informazioni:

- Totale righe lette, righe valide e righe non valide.
- Conteggio dei messaggi suddiviso per ciascuno dei tre livelli.
- Per i soli messaggi di livello **ERROR**, memorizzare il primo e l'ultimo timestamp incontrati (i log sono memorizzati in ordine cronologico).
- Contare quante volte nei messaggi delle righe valide compare (come sottostringa case-sensitive) la parola "memory" o la parola "connection".

PUNTO E: Scrittura del Report: Scrivere nel file di report (usando `fprintf`) le statistiche calcolate. Se non sono presenti eventi **ERROR**, stampare N/D per i timestamp di errore.

Al termine dell'elaborazione, stampare su `stdout`:

`Analisi completata. Report scritto correttamente.`

e terminare con exit status 0.

Output di Controllo (Esempi di Esecuzione)

Caso 1 — Scansione di un log valido

Supponendo che il file `input.log` contenga:

```
INFO;2026-07-08T09:30:00;Server started
WARN;2026-07-08T09:45:10;High memory usage
ERROR;2026-07-08T10:01:22;Database connection failed
INFO;2026-07-08T10:05:00;User login completed
ERROR;2026-07-08T11:45:00;Memory allocation failure
```

Invocando il comando:

```
./logcli input.log report.txt
```

il programma deve stampare a schermo:

```
Analisi completata. Report scritto correttamente.
```

e scrivere nel file `report.txt` il seguente report:

```
Report Analisi Log
Totale righe: 5
Righe valide: 5
Righe non valide: 0
INFO: 2
WARN: 1
ERROR: 2
Primo ERROR: 2026-07-08T10:01:22
Ultimo ERROR: 2026-07-08T11:45:00
Occorrenze keyword memory: 1
Occorrenze keyword connection: 1
```

(Exit code atteso: 0)

Caso 2 — Invocazione con numero di argomenti errato

Invocando:

```
./logcli input.log
```

il programma deve stampare su `stdout`:

```
Errore: uso corretto ./logcli input.log report.txt
```

(Exit code atteso: 1)

Caso 3 — Presenza di righe malformate

Supponendo che `input.log` contenga:

```
INFO;2026-07-08T09:30:00;Server started
RIGA_MALFORMATATA
ERROR;2026-07-08T10:01:22;Database connection failed
```

Invocando il comando, il report generato `report.txt` dovrà essere:

```
Report Analisi Log
Totale righe: 3
Righe valide: 2
Righe non valide: 1
INFO: 1
WARN: 0
ERROR: 1
Primo ERROR: 2026-07-08T10:01:22
Ultimo ERROR: 2026-07-08T10:01:22
Occorrenze keyword memory: 0
Occorrenze keyword connection: 1
```

(Exit code atteso: 0)