

Sviluppare un'applicazione in C che generi un binario denominato `counter` che simula in parte il comando `wc` di bash: effettua un conteggio dei caratteri totali (il programma deve accettare l'opzione `-c`) o delle righe (il programma deve accettare l'opzione `-l`). Il delimitatore di riga da considerare è `"\n"`. Il programma deve funzionare in due modalità: "modalità piping" con i dati che arrivano dal canale `stdin` oppure "modalità diretta" con i dati che arrivano da un file passato come argomento. Il risultato è un conteggio (un numero intero o una coppia di numeri separati da uno spazio oppure una stringa vuota in caso di errore): se usate insieme le opzioni ("`-l`" e "`-c`") lavorano entrambe e l'output presenta due valori separati da spazio nell'ordine delle stesse opzioni. Restituire un valore di uscita pari a 0 in caso di successo, maggiore altrimenti (input non corretto, errori di sistema, file non esistente, altri problemi ...).

Si vorrebbe poi distinguere l'output tra `stdout` (i valori calcolati) e `stderr` (messaggi di errore eventuali) e in più far sì che il processo principale, dopo aver verificato la correttezza della sintassi, demandi a un sottoprocesso "figlio" il compito della lettura dei dati e della stampa finale. In caso d'errore si deve restituire un valore maggiore di 0 che sia poi visualizzabile da bash nella variabile `$?` .

Esempi d'uso:

```
counter -l nomefile.txt # 10      (stdout) |                (vuoto stderr) [diretta]
ls /tmp | counter -l      # 7      (stdout) |                (vuoto stderr) [piping]
ls /tmp | counter -c -l # 30 7    (stdout) |                (vuoto stderr) [piping]
ls /tmp | counter -l -c # 7 30    (stdout) |                (vuoto stderr) [piping]
ls /tmp | counter -x      #        (vuoto stdout) | ?Opzione errata (stderr) [piping]
```

(i valori sono gli stessi del comando "`wc`" con input "`plain-text`". Ad esempio l'output di `ls /tmp | counter -c` deve essere uguale a `ls /tmp | wc -c` e così anche per l'opzione `-l`)

Creare un file "`compile.txt`" contenente il comando usato per la compilazione che deve essere del tipo `gcc ... -o counter` (dove `...` è un singolo file sorgente o una lista di files)

Attività:

- Implementazione generale della modalità *diretta* [punti: 5]
- Implementazione generale della modalità *piping* [punti: 5]
- Separazione dei flussi `stdout` e `stderr` e generazione sottoprocesso con codice di uscita 0 (ok) o maggiore di zero (errore) visualizzabile da shell [punti: 5]
- Impostazione di un makefile corretto per la compilazione [punti: 5]
(compila il/i sorgente/i solo se necessario: il file deve compilarsi correttamente utilizzando semplicemente il comando `make` senza parametri)