Programma in C

1. [3] Creare un'applicazione *PostOffice* che accetti come argomenti tre parametri: un intero **<n>** tra 1 e 10 (compresi), il percorso completo di un file esistente ***</path/to/file.txt>***, un pid di un processo esistente **<pidInput>**. Per esempio, l'applicazione può essere lanciata con

   "`./PostOffice 10 /tmp/ciao.txt 1450`"

   con `<n>=10` e `<pidInput>=1450`. L'applicazione deve correttamente gestire gli errori sui parametri: numero parametri errato, `<n>` non valido, file non esistente e `<pidInput>` non valido. Ogni errore deve stampare un messaggio di errore su stderr ed il programma deve terminare con un codice di errore.

2. [4] PostOffice deve gestire la creazione di **<n>** lavoratori, ognuno identificato da un processo separato. Al momento della propria creazione, ogni lavoratore deve inviare un segnale **SIGTERM** esclusivamente al processo con PID **<pidInput>**.
3. [5] Ogni lavoratore (non PostOffice!) dovrà gestire l'arrivo di segnali **SIGUSR1** e **SIGUSR2**. Nel dettaglio:
   a. il lavoratore deve supportare la ricezione di un singolo segnale **SIGUSR1**, rispedendolo al mittente. Ogni successivo **SIGUSR1** dovrà essere gestito con il comportamento di default (far terminare il lavoratore).
   b. Il lavoratore deve gestire molteplici **SIGUSR2**, restituendoli al mittente.

Fino a questo momento PostOffice ed i lavoratori devono rimanere in attesa e rispondere ai segnali.

4. [5] All'arrivo di un segnale SIGWINCH (28), l'obiettivo è quello di trascrivere su una coda (message queue) il contenuto del file ***</path/to/file.txt>*** ricevuto in input, riga per riga. La coda deve **essere creata** (permessi 0777), usando una chiave generata con parametri il file ricevuto e il PID del processo principale, PostOffice. Il tipo dei messaggi inviati è ininfluente per questo punto (ma lo diventa dai prossimi). Per esempio, se il file ha 3 righe, dovranno essere inviati 3 messaggi, ognuno con il contenuto di una riga.

   NB: le righe del file contengono una sola parola e NON deve essere inviato sulla coda il carattere di nuova linea, mentre deve essere inviato il carattere di terminazione (devono essere stringhe!).

5. [6] I messaggi sulla coda devono essere inviati solo dai lavoratori usando come tipo del messaggio il loro PID. **NB:** la lettura del file può avvenire nei singoli lavoratori o nel padre (attenzione all'eventuale sincronizzazione!).
6. [3] Quando viene rilevata la fine del file ed il contenuto intero è stato trascritto, tutti i lavoratori devono essere terminati, e a seguire il PostOffice deve terminare.
7. [5] La scrittura sulla coda deve avvenire in maniera sincronizzata ed equa tra i processi, con un'attesa di 1 secondo tra una scrittura ed un'altra (solo la prima scrittura è immediata).

   Esempio, se abbiamo 5 lavoratori e 10 righe di testo:

   a. La trascrizione dei 10 messaggi finirà dopo 9 secondi (9 attese di 1 secondo, con il primo messaggio inviato subito).
   b. Ogni processo scriverà al massimo 2 messaggi (scrittura equa!)
   c. La scrittura avverrà a turni

NB: tutti i parametri, stringhe o altre variabili testuali avranno come lunghezza massima 255 bytes.