Message Dispatcher

Creare un makefile per la generazione di un eseguibile “msgDispatcher.out”. Tale programma dovrà gestire la comunicazione con dei processi figli creati dinamicamente. In particolare, il programma dovrà accettare da console dei comandi, i quali potranno essere stringhe (es: “ciao come stai”) o interi>0 (es: “32”). Se il comando equivale ad un intero > 0 allora il programma dovrà creare un nuovo figlio (fino ad un massimo di 5). Se il comando contiene una stringa allora essa dovrà essere salvata in memoria.
Il programma dovrà poi supportare la ricezione di due segnali: SIGUSR1 e SIGUSR2. Alla ricezione di uno dei due segnali il programma dovrà delegare un thread per l’invio di un messaggio a tutti i figli attivi. Il messaggio, che dovrà essere inviato attraverso pipes anonime, dovrà contenere l’id del figlio a cui viene mandato, l’indice *i* (dove i = 0 se il figlio è stato il primo ad essere creato, i=5 il figlio è l’ultimo ad essere stato creato), e l’ultima stringa ricevuta dall’utente tramite comando.
Ogni figlio, una volta ricevuto il messaggio personalizzato, dovrà stamparlo e terminare, lasciando dunque al programma la possibilità di generare ulteriori figli (reset dell’indice *i*).
Ogni thread dovrà inoltre documentare in un file di log (/tmp/log.txt) il numero del segnale che ha portato alla sua generazione ed eventuali altre operazioni svolte.

Nel particolare il programma dovrà implementare le seguenti funzionalità

- Operazioni di routine:
    - Stampare il proprio PID come prima riga
    - Gestire la ricezione di comandi su stdin, interi > 0 o stringhe.
    - Gestire la chiusura con CTRL+C per terminare i figli attivi.
- Gestione segnali:
    - Impostare la ricezione dei segnali SIGUSR1 e SIGUSR2
- Gestione Thread
    - Generare un nuovo thread ad ogni nuovo segnale (SIGUSR1/2)
    - Scrivere su un file di log le operazioni del thread
    - Invio ai figli attivi di un messaggio (con pipe anonime) contenente l’ultima stringa inviata dall’utente, l’indice del figlio ed il suo pid.
    - Terminare il thread a fine invio
- Gestione figli:
    - Generare un nuovo figlio ogni volta che si riceve un comando intero > 0
    - Gestire la ricezione di messaggi su pipe anonime
    - Ogni figlio dovrà terminare dopo la ricezione e stampa del messaggio ricevuto
    - Consentire l’esistenza contemporanea di un numero MAX_CHD massimo di figli

Esempio:

```
michele@FISSO:~/OS/secSim$ make
Built all binaries
michele@FISSO:~/OS/secSim$ ./msgDispatcher.out
[MAIN] my id is 1186
43
[CHD] I'm a new child with id 1187, and I'm waiting for msg from my father
23423
[CHD] I'm a new child with id 1188, and I'm waiting for msg from my father
2
[CHD] I'm a new child with id 1189, and I'm waiting for msg from my father
cioa come va
[MAIN] Msg saved
hey come va
[MAIN] Msg saved
[CHD] I received the following message: To Child 1 PID 1188: hey come va
[CHD] I received the following message: To Child 0 PID 1187: hey come va
[CHD] I received the following message: To Child 2 PID 1189: hey come va
43
[CHD] I'm a new child with id 1191, and I'm waiting for msg from my father
2
[CHD] I'm a new child with id 1192, and I'm waiting for msg from my father
4
[CHD] I'm a new child with id 1193, and I'm waiting for msg from my father
5
[CHD] I'm a new child with id 1194, and I'm waiting for msg from my father
3
[CHD] I'm a new child with id 1195, and I'm waiting for msg from my father
2
[MAIN] Too many children. Send msg to free them
^C

michele@FISSO:~/OS/secSim$
```

```
michele@FISSO:/mnt/c/Users/miki4$ kill -10 1186
```

```
michele@FISSO:/mnt/c/Users/miki4$ cat /tmp/log.txt
[THREAD] I'm a new thread generated by signal 10
[THREAD] Sending message to children
[THREAD] children reset
[THREAD] Terminating thread
```