Pipeline di code, con filtraggio

Si realizzi in linguaggio C/C++ un'applicazione **multiprocesso** basata su **monitor signal-and-continue** e **code di messaggi UNIX**, secondo lo schema in **pipeline** illustrato di seguito e in figura.

Il **generatore** include 4 processi in totale che dovranno implementare un problema **produttore-consumatore con coda circolare** (dimensione della coda pari a 4). In particolare, 2 processi (P) fungeranno da produttori di messaggi e 2 processi (C) fungeranno da consumatori di messaggi. Il messaggio generato dai produttori deve contenere i seguenti campi:

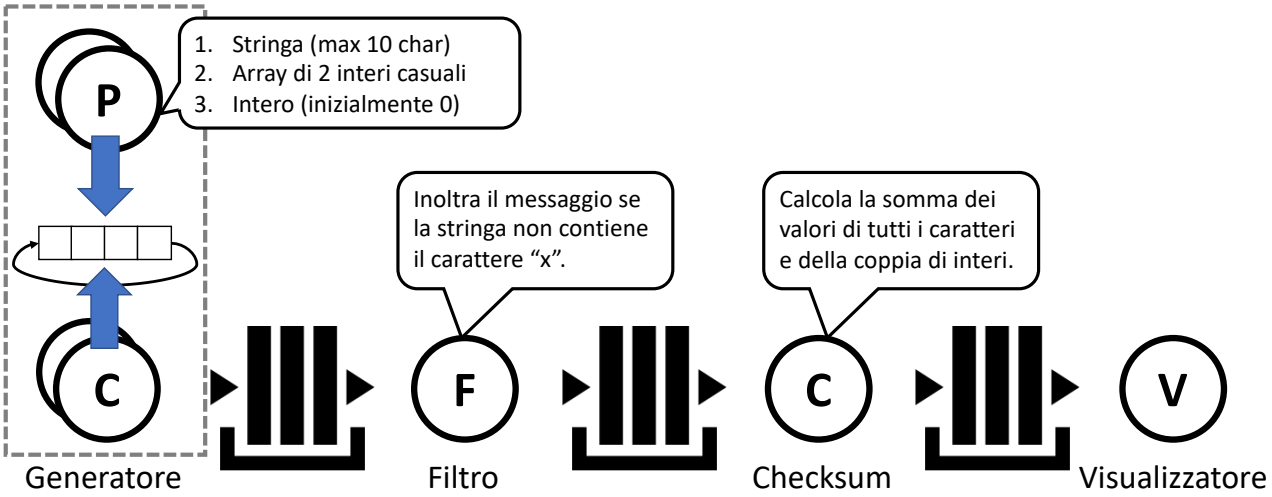
1. una stringa di massimo 10 caratteri;
2. un array di 2 interi;
3. una variabile intera.

I processi consumatori invieranno il messaggio consumato sulla coda di messaggi verso il processo *filtro* (vedere figura). La stringa dovrà essere generata utilizzando caratteri casuali[1]; i valori dell'array di interi dovrà essere scelto casualmente tra 0 e 9; la terza variabile intera dovrà essere posta a 0. Ogni processo produttore dovrà produrre 4 messaggi. Dopo aver inviato in totale 8 messaggi (4 per ogni consumatore), i processi consumatori dovranno terminare. **Produttori e consumatori dovranno essere disciplinati utilizzando il costrutto monitor con strategia signal-and-continue.**

Il processo **filtro** dovrà ricevere i messaggi dai processi generatori (i.e., i consumatori), e dovrà effettuare una ricerca del carattere 'x' all'interno della stringa[2]. Se la stringa non contiene il carattere 'x', allora il messaggio viene inoltrato al processo **checksum**; altrimenti, il messaggio viene ignorato. Il processo filtro dovrà ripetere 8 volte la ricezione e la ricerca, e poi terminare.

Il processo **checksum** dovrà ricevere i messaggi dal processo filtro. Ad ogni messaggio, dovrà calcolare la somma di tutti i caratteri della stringa e del vettore di 2 interi, e inserire il risultato nella terza variabile intera del messaggio. Infine, dovrà inviare il messaggio risultante al processo **visualizzatore**, il quale farà una stampa a video di tutto il messaggio. Entrambi i processi terminano dopo aver elaborato 8 messaggi.

Assumere che i processi **checksum** e **visualizzatore** possano non elaborare tutti e 8 messaggi generati, terminando se non dovessero trovare nessun messaggio nelle rispettive code.



[1]: Si assegnino alla stringa valori casuali tramite: `msg.stringa[i] = 'a' + (rand() % 26)`. Si ricordi di assegnare all'ultimo carattere della stringa il carattere terminatore (valore 0),
[2]: Per effettuare la ricerca del carattere, è possibile utilizzare: `strchr(msg.stringa, 'x');` la funzione `strchr` ritorna NULL se il carattere non è presente nella stringa.