Esercizio con select

Si progetti un'applicazione distribuita Client/Server che consenta di gestire i conti corrente di una banca. Il Server deve fornire due tipi di servizio: la gestione di movimenti (versamenti e prelievi) e la visualizzazione di informazioni sul conto (estratto conto completo inteso come insieme di tutti i movimenti).

L'applicazione prevede l'implementazione di un Server e di due Client.

Il Server prevede una fase iniziale di creazione di un certo numero di conti su cui lavorare, il cui codice viene deciso inizialmente e che i clienti devono conoscere.
Il Server (*Server_banca.c*) deve accettare le richieste di operazione, discriminando tra i due tipi di operazioni che vengono ricevute attraverso socket diverse. Obbligatoriamente si usi la primitiva *select* per distinguere le richieste di versamento/prelievo, da gestire in maniera sequenziale, e quelle di visualizzazione dell'estratto conto, che si devono gestire in modo parallelo una volta ricevute, vista la probabile maggiore durata.

Il primo Client (*Client_movimenti.c*) effettua solo richieste di versamento/prelievo su un conto specificato e ha l'interfaccia di invocazione

movimenti NomeHost

Il secondo Client (*Client_estrattoConto.c*) effettua solo richieste di visualizzazione dell'estratto conto e viene invocato con:

estrattoConto NomeHost

dove NomeHost è il nome dell'host su cui è in esecuzione il Server.
I due Client inoltrano le richieste ciascuno in modo appropriato al Server. Per ogni possibile operazione si preveda anche una gestione di eventuali condizioni anomale (come il numero di conto sbagliato o altre condizioni di errore).
Entrambi i Client sono implementati come processi ciclici che continuano a fare richieste sincrone fino ad esaurire tutte le richieste utente (fino alla fine del file di ingresso dell'utente).

Con maggiore dettaglio sulla implementazione, il primo Client ciclicamente legge il numero di conto (a default da tastiera), il tipo di operazione e l'importo dell'operazione e richiede la operazione attendendone il risultato o trattando le opportune condizioni anomale.
Ad esempio, la interazione di I/O potrebbe essere:

- Numero conto corrente: 123
- Operazione (V=versamento, P=prelievo): V
- Importo: 100
- Operazione effettuata correttamente

ecc.
In caso di prelievo superiore all'importo disponibile il Server deve segnalarlo al Client, che a sua volta lo notifica all'utente.

Il secondo Client legge ciclicamente il numero del conto e il Server risponde con un file di testo (uno per ogni conto, il cui nome coincide con il numero del conto) che contiene una riga per ogni movimento effettuato sul conto. Il file viene visualizzato sulla console dell'utente. Ovviamente, se il parametro fornito non corrisponde ai conti attivi nel Server, il Client deve segnalare l'errore. Per esempio:

- Numero conto corrente: 121
- Operazione effettuata: file:

// Seguono movimenti…uno per riga
…
//…

- Numero conto corrente: 999
- Errore: numero di conto inesistente