myFinger.c

Introduzione

Il progetto è diviso in due grandi funzioni: **main** e **entryAnalizer**.
La prima si occupa della catalogazione degli input e della selezione degli utenti di cui finger deve dare informazioni, la seconda si occupa di recuperare queste informazioni e formattarle per la stampa a schermo.
Sono poi presenti alcune funzioni di appoggio che verranno brevemente trattate alla fine.

---

main

Possiamo suddividere la funzione main a sua volta in quattro grandi blocchi: *catalogazione input*, *ordinamento alfabetico*, *recupero degli utenti* e chiamata a *entryAnalizer*.

La catalogazione avviene attraverso una semplice iterazione negli elementi di argv, questi vengono infatti comparati con le opzioni possibili e, in caso di corrispondenza, i valori booleani di queste opzioni vengono attivati (ad esempio -l o -p).
Se la corrispondenza non è rilevata l'argomento viene salvato come possibile utente da ricercare (nello userArray), impostando a true la variabile booleana search.

L'ordinamento alfabetico lavora proprio sullo userArray, questo viene infatti letto e attraverso strtok suddiviso in un array di stringhe, il quale verrà passato in input a qsort.

La parte più corposa è sicuramente **il recupero degli utenti**, questo è infatti suddiviso in due modalità: *ricerca e non*.
Il recupero più semplice è quello privo di ricerca, per ottenere gli utenti è infatti sufficiente scorrere l'utmp file filtrando le user entry. Questo poiché in utmp si trovano solo utenti loggati e dunque si ottiene subito la lista da analizzare.
Per quanto riguarda la ricerca la questione è più complessa, è necessario infatti valutare tre situazioni distinte:

- il nome inserito corrisponde al nome di login
- il nome inserito corrisponde a una parte (o tutto) il nome reale
- il nome inserito corrisponde al nome di login di un utente non loggato

Le prime due problematiche sono risolte attraverso due approcci molto simili: in entrambi i casi *sia il nome inserito che il nome* (login o reale) *registrato vengono trasposti in lowercase*, al fine di poterli comparare senza curarsi di maiuscole e minuscole (la ricerca di finger non è infatti case-sensitive). Per quanto riguarda però il nome reale il confronto viene ripetuto su ogni parola componente l'intero nome (dunque nome o cognome o anche più nomi).
L'ultima situazione, l'utente non loggato, sfrutta un approccio diverso: *si richiama la funzione getpwnam,* questa infatti controlla il file /etc/passwd che, anche se con informazioni minori, mantiene i dati di tutti gli utenti, loggati e non.

<<data la differenza di informazioni reperibili e operazioni permesse fra utenti loggati e non, vengono inserite alla fine dello username (nella stringa contenente gli utenti) la lettera Y o N per distinguerli>>

L'ultima sezione del main è la chiamata alla funzione **entryAnalizer**, qui tutto ciò che si fa è stampare eventuali header (come con l'opzione -s selezionata), suddividere attraverso strtok la stringa contenente gli utenti da mostrare e impostare la variabile booleana logged a true o false a seconda dei casi, per permettere il corretto funzionamento di entryAnalizer.

---

entryAnalizer

Si occupa dell'effettivo recupero delle informazioni e della loro formattazione. Consiste in una catena di azioni raggruppabili in: **recupero delle entry** nel file pwd e utmp (se utente loggato), recupero delle informazioni contenute nel **gecos** (nome, office, etc.), calcolo **idle time** e recupero della **data di login**, recupero dei dati sulle **mail**, recupero dei file **plan**, **project** e **pgpkey** e infine print formattato.

<<Essendo il recupero delle entry abbastanza scontato e l'analisi del gecos semplicemente una tokenizzazione possiamo concentrarci direttamente sull'idle time e il recupero della data di login>>

Per quanto riguarda **l'idle** la prima azione svolta è il calcolo della differenza fra il tempo attuale e il tempo di creazione del file tty, questo poi, essendo salvato in formato epoch, viene formattato in data e ora attraverso semplici calcoli di divisione e modulo.
Per **la data di login** il recupero è meno elaborato, si sfrutta la funzione strftime in combinazione con i dati contenuti nella entry del file.

Il recupero dei dati sulle **mail** consiste nel ottenere le ultime date di lettura e modifica del file mail (corrispondono infatti all'ultima lettura della posta e all'ultima mail in entrata), questo si ottiene attraverso la struct stat applicata al file /var/spool/mail, strftime per la formattazione delle date e un semplice controllo di antecedenza tra i due valori per dedurre se ci sono mail non lette.

I file **plan**, **project** e **pgpkey**, sono anch'essi abbastanza rapidi da ottenere, basta controllarne l'esistenza attraverso open e nel caso salvarne l'fd per poi eseguirne il print in seguito.
Nel programma l'azione è specificata in una funzione d'appoggio chiamata fileRetrieve.

Per quanto riguarda il **print** abbiamo due possibilità: *formattazione -l o -s*, in entrambi i casi tutto ciò che viene fatto è eseguire il layout di finger attraverso vari print e il calcolo della lunghezza massima raggiunta dagli username e dai realname al fine dell'incolonnamento.

Funzioni d'appoggio

Le funzioni d'appoggio presenti sono: **stringCompare**, **compare**, **strToLower**, **filePrinter**, **fileRetrieve** e **utEntryRetrieve**.

Le prime due sono in realtà strettamente connesse, poiché vengono definite per essere passate a qsort come funzioni necessarie a decretare l'ordinamento, nello specifico viene chiamata compare che richiama a sua volta stringCompare.

Poi d'interesse sono anche filePrinter e fileRetrieve, la prima si occupa di formattare bene il print dei file per evitare situazioni di doppio \n o totale assenza, la seconda invece controlla prima di tutto se è possibile accedere a un certo file e in seguito lo carica su un puntatore.

L'ultima, anch'essa molto semplice, è utEntryRetrieve che si occupa di scorrere il file utmp alla ricerca di corrispondenza con un certo username passato in input.