Homework 1

Introduzione

L'homework 1 consiste nella scrittura di un programma in C che simula il comando finger di Linux.
A differenza del comando originale, questa simulazione non contiene l'implementazione del lookup delle informazioni per utenti remoti.

Dettagli sul comando sono disponibili alla seguente pagina https://linux.die.net/man/1/finger.

Progetto

Il progetto è composto dalle seguenti parti:

- finger che contiene il main dell'applicazione
- parser che si occupa della lettura dei parametri e l'inizializzazione e popolamento delle strutture dati
- printer che si occupa della fase di scrittura dell'output

Strutture dati

Le strutture dati adoperate sono definite nel file **`structure.h`** e di seguito indicate nel dettaglio.

La struttura **`format_t`** contiene informazioni utili in fase di scrittura dell'output. Presenta …

La struttura **idletime_t** rappresenta il tempo di inattività espresso in ore, minuti e secondi. L'utilizzo di questa struttura è utile nella fase di output, dove il formato della stringa può variare.
Infatti, ad esempio, la stampa su righe multiple prevede di:

- stampare la stringa "*X hours Y minutes idle*" se l'ora è maggiore di zero
- stampare la stringa "*X minutes Y seconds idle*" altrimenti

La struttura **`user_t`** contiene le informazioni su un utente. Un utente contiene le seguenti informazioni:

- il nome utente
- il realname dell'utente
- il nome del terminale e il suo suffisso
- la home directory
- la shell
- il tempo di inattività
- informazioni GECOS parziali:
    - ufficio
    - numero di telefono dell'ufficio
    - numero di telefono di casa

Un utente può essere sia un utente che ha effettuato una login che un utente di sistema.

La struttura **`finger_t`** è la struttura principale del progetto. Essa contiene in particolare:

- un array dinamico degli utenti
- un array dinamico degli utenti univoci (senza ripetizioni)

Si tiene traccia degli utenti univoci per facilitare la stampa in caso di formato multilinea per cui un medesimo utente sia loggato più volte.

Finger

Il finger (implementazione in finger.c) è il punto di partenza del progetto (infatti contiene il metodo **`main`**).
Esso delega le seguenti attività:

- chiama il parser per leggere i parametri forniti
- chiama il printer per gestire la fase di output

Si occupa inoltre di allocare la memoria iniziale e di pulire la memoria utilizzata in fase di terminazione del programma.

Il file finger.h contiene i prototipi delle funzioni ed eventuali dichiarazioni di variabili e costanti.
Il file finger.c contiene la logica implementativa delle funzioni dichiarate nel file finger.h.

Parser

Il parser (implementazione in parser.c) si preoccupa di leggere i parametri passati in ingresso.

Le attività generali svolte dal parser sono le seguenti:

- lettura dei parametri
- popolamento delle strutture dati

In particolare, si preoccupa di stabilire il tipo di formato atteso dal programma (se singola riga o multilinea).

Inoltre, legge gli utenti passati come argomento e, per ognuno, arricchisce l'apposita struttura dati con le sue informazioni.

In caso di nessun utente specificato come argomento, considera l'utente attualmente loggato.

Inoltre, verifica che tutti i parametri siano validi. In caso negativo, stampa l'opzione non valida e restituisce un errore.
Ad esempio, se passiamo l'opzione inesistente **-k**, otteniamo il seguente errore:

```
francescopio@francescopio:~/homework1/finger$ ./finger -k
finger: invalid option -- 'k'
finger: usage: finger [-lmsp] [user ...]
```

Il file parser.h contiene i prototipi delle funzioni ed eventuali dichiarazioni di variabili e costanti.
Il file parser.c contiene la logica implementativa delle funzioni dichiarate nel file parser.h.

Printer

Il printer (implementazione in printer.c) consiste nella fase finale del programma. Il suo compito principale è quello di stampare le informazioni nel formato richiesto (ad esempio singola riga o multilinea).

Il file printer.h contiene i prototipi delle funzioni ed eventuali dichiarazioni di variabili e costanti.
Il file printer.c contiene la logica implementativa delle funzioni dichiarate nel file printer.h.

Startup

Build

La fase di compilazione e linking è automatizzata mediante l'uso di Make.
Quindi vi è la presenza del file Makefile che definisce:

- compilazione di parser basandosi su parser.c e parser.h
- compilazione di printer basandosi su printer.c e printer.h
- compilazione di finger basandosi su finger.c e finger.h
- linking di tutti i file e creazione dello script **`finger`**

Per eseguire la build del programma posizionarsi nella cartella dei sorgenti ed eseguire il comando **`make`** come mostrato di seguito:

```
francescopio@francescopio:~/homework1/finger$ make
cc -c finger.c
cc -c parser.c
cc -c printer.c
cc -o finger finger.o parser.o printer.o
```

Run

Per eseguire il programma è necessario posizionarsi nella cartella dei sorgenti ed eseguire lo script denominato **`finger`** nel seguente modo:

```
francescopio@francescopio:~/homework1/finger$ ./finger francescopio
Login: francescopio                     Name: francescopio
Directory: /home/francescopio           Shell: /bin/bash
Office: 12, x1234                       Home Phone: x5678
On since Fri Jun 28 08:26 (CEST) on tty7 from :0
        7 hours 30 minutes idle
Mail forwarded to Test .forward file Row 1
Test .forward file Row 2
Mail last read Thu Jun 27 09:12 2024 (CEST)
PGP key:
Test .pgpkey file row 1
Test .pgpkey file row 2
Project:
Test .project file row 1
Test .project file row 2
Plan:
Test .plan file Row 1
Test .plan file Row 2
```