# Relazione su `Myfinger`

## 1. Obiettivo del progetto

L'obiettivo principale è stato sviluppare un programma in linguaggio C che emulasse il comando `finger`, fornendo informazioni sugli utenti locali del sistema. Non è stato implementato il supporto per utenti remoti, come richiesto dalla consegna.

## 2. Vincoli e requisiti

### Vincoli tecnici

1. Non era permesso utilizzare funzioni della famiglia `exec*(3)`.

### Requisiti funzionali

1. Restituire informazioni sugli utenti locali, come nome, terminale, orario di accesso, directory personale e shell.
2. Gestire correttamente le opzioni specificate dall'utente e i nomi utente passati come argomenti.

## 3. Struttura del codice

Il codice è suddiviso in più funzioni per modularità e leggibilità. Di seguito è riportata una panoramica delle principali sezioni.

### 3.1 Funzioni principali

- **`main`**:
    - Analizza gli argomenti della riga di comando, identificando opzioni (`-l`, `-s`, `-m`, `-p`) e nomi utente specifici.
    - Gestisce due casi: nessuna opzione specificata o opzioni/nome utente specificati.
    - Invoca funzioni per selezionare gli utenti (`userSelection`) e processare le opzioni (`optionSelection`).
- **`optionSelection`**:
    - Elabora le opzioni passate dall'utente e determina quale funzione di stampa (`printS` o `printL`) invocare.
    - Gestisce errori di inserimento opzioni.
- **`userSelection`**:
    - Recupera automaticamente l'elenco degli utenti connessi se non sono specificati nomi utente.
    - Verifica la validità dei nomi utente forniti come argomenti confrontandoli con l'elenco degli utenti nel sistema.
- **`gecos_format`**:

    - Suddivide il campo `gecos` della struttura `passwd` per estrarre informazioni come nome completo, ufficio, telefono, ecc.

- **`printS`**:
    - Implementa una visualizzazione "ridotta" delle informazioni utente, come login, nome reale, terminale, orario di accesso e host di accesso.
    - Utilizza le strutture `passwd` e `utmp` per recuperare i dati degli utenti.

- **`printL`**:
    - Implementa una visualizzazione "dettagliata" delle informazioni, includendo directory personale, shell, numeri di telefono e informazioni sui file `.plan`, `.project` e `.pgpkey`.
    - Controlla la presenza di file specifici nella directory personale dell'utente.

# 4. Soluzioni implementative

## Gestione della riga di comando

Il programma analizza argomenti e opzioni con un loop, distinguendo tra nomi utente e opzioni.

## Gestione dinamica della memoria

L'uso di `malloc`, `realloc` e `strdup` consente di gestire array dinamici di stringhe per nomi utente.

- **Accesso ai dati del sistema**:
  Le strutture `passwd` e `utmp` sono state utilizzate per ottenere informazioni sugli utenti locali.
- **Formattazione e visualizzazione**:
  I dati sono stati presentati in un formato leggibile, simile al comando `finger`, con un'attenzione particolare alla compatibilità con le opzioni specificate dall'utente.