
C Programming Assignment: "এসো টাইপিং শিখি"

Objective:

Develop a terminal-based typing speed test game in C using the **ncurses** library. The game should generate a typing passage based on a pool of words stored in a file (containing words line-by-line). The difficulty of the game will affect the passage length and complexity. The game will highlight correct and incorrect keystrokes, show performance statistics, and stop after a set time, displaying the results.

Requirements:

1. **Game Mechanics:**
    - Upon starting, the game will generate a passage of text using words from a word pool. This pool will be provided in a file.
    - The difficulty level will determine the length and complexity of the generated passage.
        - **Easy**: Shorter passages with common, simple words.
        - **Medium**: Longer passages with moderately difficult words.
        - **Hard**: Long, complex passages with challenging words.
    - The words for the passages will be randomly selected from the file of words.
    - Display the typed passage at the top of the screen, and the user's typed characters below.
    - As the user types, their characters should be highlighted. Correct characters appear in **green**, and incorrect ones in **red**.
    - The game should time the user's typing session, with a time limit (e.g., 60 seconds).
    - Once the time expires, the game will stop and show the statistics, including WPM (Words per minute), accuracy, and other relevant performance data.

2. **Word Pool File:**
    - The word pool should be provided in a text file, where each line contains a word.
    - The student must read words from this file and generate the typing passage based on the difficulty level. For example:
        - Easy: 20 random words.
        - Medium: 40 random words.
        - Hard: 60 random words.
    - The file will be redirected into the program using input redirection (`<` or `>`).

3. **Key Features:**

- **Word Selection**: Read words from the input file and randomly select a passage for the user to type based on difficulty.
- **Progress Highlighting**: As the user types, the program will highlight:
  - Correct characters: **green**
  - Incorrect characters: **red**
- **Typing Statistics**:
  - **Words per Minute (WPM)**: Number of correct words typed per minute.
  - **Accuracy**: Percentage of correctly typed characters.
  - **Total Keystrokes**: Total number of characters typed, including correct and incorrect characters.
  - **Mistakes**: Count of incorrect characters typed.
  - **Characters per Second (CPS)**: Number of characters typed per second.
  - \

4. **Statistics Formulas:**

   - **Words per Minute (WPM)**: $WPM = \frac{Total\ Correct\ Words}{Time\ Elapsed\ in\ Minutes}$

   - **Accuracy**: $Accuracy = (\frac{Correct\ Characters}{Total\ Characters}) * 100.0$

   - **Characters per Second (CPS)**: $CPS = \frac{Total\ Characters}{Time\ Elapsed\ in\ Seconds}$

5. **Defining Word Complexity:**
   - Easy Words:
     - Length: 5 to 7 letters
     - Syllables: Typically 2 to 3 syllables
     - Examples: Cat, dog
   - Medium Words:
     - Length: 1 to 4 letters
     - Syllables: Typically 1 or 2 syllables
     - Examples: Table, Flower
   - Hard Words:
     - Length: 8+ letters
     - Syllables: 3 or more syllables
     - Examples: rhinoceros, perseverance

6. **Interface Design**:

   - Use **ncurses** to manage the terminal-based interface.
   - Display the passage to be typed at the top, the user's input below, and the stats in the footer once the game ends.
   - Ensure the interface is clean and easy to understand.
7. **Input/Output Handling**:

- Use input redirection to read the word pool file, e.g., `./typing_game < word_pool.txt`.
- The program should handle incorrect key presses gracefully by marking incorrect characters in red and continuing the game.

Constraints:

- The students are not allowed to read/write files, so input redirection (`<`) will be used to provide the word pool file.
- Ensure that the user can start the game by pressing **Enter** and can quit at any point by pressing **Esc**.
- The program should handle basic edge cases, such as pressing incorrect keys, and should continue without crashing.

Grading Rubric:

Criterion	Max Points	Description
Functionality	40	The game successfully reads from the input file, generates a typing passage based on difficulty, tracks user input, and calculates WPM, accuracy, and other stats.
Keystroke Handling	10	The program handles keystrokes properly, highlighting correct (green) and incorrect (red) characters and keeping track of mistakes.
Word Pool Management	10	The program successfully reads the word pool from the redirected input file and generates a passage based on the selected difficulty.

Timing Mechanism	10	The game implements a timer that counts down and stops after the designated time limit (e.g., 60 seconds).
User Interface (ncurses)	10	The user interface is neat and clear, displaying the passage to type, user's typed input, progress, and results using ncurses.
Accuracy and Statistics Calculation	10	The game correctly calculates and displays statistics like WPM, accuracy, and CPS based on the provided formulas.
Code Structure and Readability	10	The code is well-organized, modular, and properly commented, making it easy to understand and maintain.
Bonus: Restart/Exit Option	5	The game allows the user to restart or exit the game at any point with a clean interface.
Bonus: Handling Input Edge Cases	5	The program gracefully handles edge cases, such as pressing backspace, entering incorrect characters, handling unexpected input, or typing all words before timeout.

Bonus Points:

- **Restart/Exit Option (5 points)**: Implement a restart or exit option to provide the user a smooth experience.
- **Handling Edge Cases (5 points)**: Handle cases like backspace, invalid input, and unexpected keypresses gracefully, continuing the game without errors.

Extra Notes:

- **Libraries to Use**: `ncurses` for terminal UI management and basic file I/O for reading the word pool file.
- **Compilation**: Compile with `gcc -o typing_game typing_game.c -lncurses` (assuming ncurses is installed).

File Input/Output: Students will use input redirection to read the word pool file: `./typing_game < word_pool.txt`

References:

- **Libraries usage example**
- **Input/Output redirection**