資工115 鄭聿喬 41147011S

執行過程與結果

從執行結果來看，程式成功完成了以下幾個步驟：

1. **檔案操作**：
   - 使用 `open()` 打開了 `foo.bar` 作為標準輸入，並以只讀模式（`O_RDONLY`）成功完成文件的打開操作。
   - 使用 `open()` 創建或覆蓋了 `result` 作為標準輸出，並以寫模式（`O_WRONLY | O_CREAT | O_TRUNC`）成功完成操作。

2. **文件描述符重定向**：
   - 使用 `dup2()` 成功將 `foo.bar` 重定向到標準輸入（`STDIN_FILENO`），確保 `wc` 程式讀取的內容來自 `foo.bar`。
   - 同樣使用 `dup2()` 將 `result` 重定向到標準輸出（`STDOUT_FILENO`），使 `wc` 的輸出被正確寫入到 `result` 文件中。

3. **命令執行**：
   - 使用 `execlp()` 成功執行了 `wc` 指令，計算 `foo.bar` 中的行數、單詞數和字符數。

4. **驗證結果**：
   - `echo "Hello World" > foo.bar` 將文字 "Hello World" 寫入文件。
   - 執行 `./redirect_wc` 後，`result` 文件包含內容 `1 2 12`，分別表示文件的行數、單詞數和字符數，與 `wc < foo.bar > result` 指令的預期行為完全一致。

系統設計分析

1. **優點**：
    - **功能實現完整**：成功模擬了 Shell 指令的行為，並通過腳本驗證了正確性。
    - **資源管理得當**：在每次使用檔案描述符後，程式正確關閉了檔案描述符，避免了資源洩漏問題。
    - **錯誤處理全面**：程式在每一步都使用了 `perror()` 提供清晰的錯誤訊息，有助於在問題發生時快速定位。

2. **設計簡潔高效**：
    - 通過使用系統呼叫，程式邏輯清晰且執行高效，直接操作底層資源避免了不必要的中間過程。

程式碼討論

檢查項目

1. **程式邏輯：**
    - 使用 `open()` 打開 `foo.bar` 以讀取。
    - 使用 `open()` 打開或創建 `result` 以寫入，並確保文件被截斷（清空內容）。
    - 使用 `dup2()` 重定向：
        - 標準輸入（`stdin`）被重定向到 `foo.bar`。
        - 標準輸出（`stdout`）被重定向到 `result`。
    - 使用 `execlp()` 執行 `wc`，成功運行。

2. **測試文件內容：**
    - `foo.bar` 中的內容是：`Hello World`（1 行，2 個單詞，12 個字符，包括換行符）。

3. **執行結果：**

- `cat result` 顯示：

  ```
  1  2 12
  ```

  - 表示：1 行、2 個單詞、12 個字符，與 `wc` 的正常行為一致。

4. **記錄過程：**
   - 使用 `script` 記錄了完整操作，包括：
     - 源碼檢視（`cat redirect_wc.c`）。
     - 編譯（`gcc -o redirect_wc redirect_wc.c`）。
     - 測試文件生成（`echo "Hello World" > foo.bar`）。
     - 執行程式（`./redirect_wc`）。
     - 結果檢視（`cat result`）。

確認符合任務需求

1. 程式碼符合題目要求：
   - 使用了 `open()`、`dup2()`、`close()` 和 `execlp()`。
2. 執行結果正確：
   - `wc` 的輸出被正確重定向到 `result`。
3. 紀錄過程完整，滿足需求。

改進建議

雖然程式功能已正確實現，但可以進一步改進：

1. **添加更多錯誤處理：**
   - 檢查 `execlp()` 返回時釋放所有資源（儘管理論上不會返回）。
2. **結果驗證：**
   - 可在程式執行後直接檢查文件大小或內容，以驗證結果是否正確。

若無需改進，則此結果已完美達到要求！

結論

整體來說，本次實作成功地模擬了 UNIX Shell 指令 `wc < foo.bar > result` 的核心功能，並且結果準確無誤，資源管理良好，符合預期。該任務展示了系統呼叫的靈活性和 Shell 背後的實現原理，對於理解 UNIX 系統的設計理念具有重要意義。未來可以進一步改進在特殊情況處理和安全性上的設計，使程式更加完善與健壯。