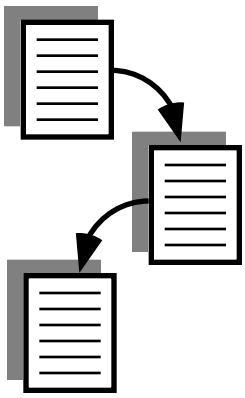
CS 1706 Spring 92
Assignment 2: Multi-File Viewer (MFV)
Due Date: February 27, 1992 in labs



Problem

Code a program to allow a user to view the top text file, (of a stack of text files), by scrolling and paging.

Purpose

This assignment will provide experience with the following coding features: 1. Separate Compilation (required); 2. Linked-Lists 3. Text-Windowing System and 4. Project Make Facilities, (not required).

Discussion

The primary viewing/browsing features of this program shall now be discussed. The program will display 20 lines of a file at a time, in rows 3-22 on the screen, (hereafter referred to as the display area). The user may browse through a MFV file by either one of three methods.

The first navigation facility is paging. This involves ‘moving’, (changing the display area), to the preceding or succeeding 20 lines in the file by pressing 'U' or 'u' / 'D' or 'd' respectively. Turbo users should use the PgUp and PgDn keys.

The ↑ ↓ cursor keys will allow the user to scroll the display area contents one line at a time, up and down respectively. When depressing either the ↑ ↓ cursor keys, the display area 20 line *window* into the MFV file will be scrolled one line backward/forward respectively. If at any point in the viewing, if the user attempts to scroll above/beyond the top/bottom of the file an error message must be inversely displayed on line 23 and the bell (control-h) sounded.

Thirdly, MFV will also allow the top or bottom of the file to be moved to directly, by the user depressing either the ‘T’ or ‘t’ / ‘B’ or ‘b’ keys, respectively. The display area should never contain blank lines for files with $\geq$20 lines. Consider a file with 100 lines, if currently lines 11 through 30 are displayed and page up is hit the first 20 lines in the file should now be on the screen. Similarly, if lines 71 through 90 are displayed and page down is hit, the last 20 lines (81..100) are then displayed. For small files with <20 lines, page up/down would receive an error message and line up/down, ↑ ↓, should not cause any line to scroll off the display. Robustness and correctness, not speed of display, should be the over-riding factors governing the implementation of the view operations.

Execution

MFV’s execution control flow will be addressed next. The program should initially present the user with a startup screen displaying the name of the program, a brief 3-4 line explanation of the program and the programmer’s name, address and telephone number. Some minor creativity in the presentation of the startup screen is desired. After either the user hits any key, or either approximately 5-7 seconds has passed, the startup screen is cleared. A brief help screen should appear, explaining the operation of the program to the user, until the user is ready to continue.

Following the brief help screen the user is prompted for the name of the initial MFV file they wish to browse. All input prompts to the user should occur on line 24. The user should be requested to wait while the file is being input from secondary memory. The linked-list for the file’s lines is now setup. At this point the viewing operations, discussed above, are at the user’s disposal to navigate through the file.

At any time during execution, a user may request that another file be opened or the current file be closed, (using the ‘O’, ‘o’ or ‘C’, ‘c’ keys respectively). When a new file is opened, (prompting occurs the same as at startup described previously), the current MFV file being viewed is ‘pushed’ onto the MFV file stack. The new file is then

read and setup for viewing, with the first 20 lines being shown in the display area. When the current file is closed the previously pushed file is automatically restored as the current file. This may be implemented in one of two ways: 1. the file may opened again, (i.e. read and setup again), with the first 'page' being displayed; 2. separate lists may be maintained for each file in the MFV stack and the current file. If the second method is used, the restored file should reopen with the same file page being displayed as when it was pushed. Turbo users should be cautious about selecting the second method due to limited memory. A user may also elect to quit MFV at any time during browsing by depressing the 'q' or 'Q' keys.

Abstract Data Type

The main data structure required for this assignment is an unordered non-circular double linked-list, to hold the text file lines, implemented with PASCAL pointers, (text book operations may be used & modified). This will permit viewing of arbitrarily long files. Additional operations should be added as necessary to compose a viewing or file window pane abstract data type.

Text Windowing

The use of a text windowing package is not required for this program, but is strongly recommended. The text windowing facilities will greatly enhance the professional *look-and-feel* of your program. This program will be used, almost unmodified, in the final course project, comprising approximately 20...25% of the final code.

Input

Sample data input files will be provided shortly for copying in the McB 116 lab.

Assumptions

No existence checking for file names is required. No file will be large enough to exhaust the heap. All lines in a file are $\leq$ 78 characters in length. All files to be viewed are in the current directory. The MFV maximum stack size is 20.

Grading

Turn in hard copies of a structure chart, source code, input/output files, program *makefile* (if employed) specifying module dependencies, and a diskette containing: ASCII source code, executable image, I/O files, and an ASCII *readme* file with execution instructions. The source code file must be named MFV.pas and the executable image MFV. The disk should contain no subdirectories and protection should be set to allow anyone access. All materials to be graded must be placed in a folder, neatly labeled, with the disk securely fastened.

In addition, your GTA will require you to demonstrate your program in the McBryde 116 PC lab during lab time. To receive partial credit for programs that are non-working, or are not fully functional, a brief one or two paragraph description of the problem(s) must be included in the assignment folder. The location, routine minimum, must also be specified along with possible corrections that need to be made.