ADVANCE MICROPROCESSOR PROJECT

DHARAMSINH DESAI UNIVERSITY

FACULTY OF TECHNOLOGY

PROJECT NAME : FILE MANAGER

PREPARED BY : PISHANG UJENIYA (IT-100)

GUIDED BY : PROF. SIDDHI JOSHI

DESCRIPTION

My project aims to implement File Manager using Buttons as Graphical User Interface

It would be consisting of following functionalities of File Management.

1. Create a File
2. View a File
3. Copy a File
4. Rename a File
5. Delete a File
6. Create a Directory
7. Remove a Directory
8. Change a Directory
9. List all in a Directory
10. Buttons Action for Above all functions

To perform the above mentioned tasks I have used mainly three Bios Interrupts, INT 10h, INT 33h, INT 21h

Basic usage of Interrupts

INT 10h is used for Screen Manipulation.

INT 33h is used for Mouse Interrupts.

INT 21h is used for File Handling and echo text.

GRAPHICS MODE

```
MOV AX, 12H                          ; set graphics mode
INT 10H
```

TEXT MODE

```
MOV AL, 03H                          ; set text mode
MOV AH, 00H
INT 10H
```

CREATING A BUTTON RECTANGLE

```
MOV CX,21D                           ; screen column

MOV DX,210D                          ; screen line

MOV AL,09H                           ; colour of pixel

MOV AH,0CH                           ; print pixel

SNA1:INT 10H

INC CX

CMP CX,155D

JNE SNA1

INC DX

MOV CX,21D

CMP DX,259D

JNE SNA1
```

DISPLAY STRING IN BUTTON

```
MOV AL,01H                ; write mode

MOV BH,0H                 ; page number

MOV BL,04H                ; text color change

MOV CX,04D                ; no of characters in string

MOV DL,7D                 ; column start

MOV DH,14D                ; row start
MOV BP,OFFSET S1
MOV AH,13H                  ; print string
INT 10H                     ; print string interrupt
```

MOUSE INITIALISATION

```
AGAIN: MOV AX,0000H
INT 33H
CMP AX,0000H                ; ax=0 mouse driver not installed
JE AGAIN
MOV AX, 0001H
INT 33H
```

MOUSE CLICK SCANNING

```
CHECK4:
MOV AX,0003H
INT 33H
CMP BX,1H
JNE CHECK4

CMP CX,486D         ; greater than or equal 486d we want
JC CHECK3           ; to next button
CMP CX,620D         ; less than 620d
JNC CHECK4
CMP DX,210D         ; greater than or equal 210d we want
JC CHECK1           ; to first button
CMP DX,260D         ; less than 260d
JNC CHECK1          ; to first button
CALL BT4            ; action if button pressed
```

CREATING A FILE

```
CREATE:

CALL SIMPLE

LEA DX, MSG2                    ; module for creating a file

        CALL DISP1

        CALL READ1              ; read name of file to be

        LEA DX, BUFFER1[2]      ; created

        MOV CX, 0

        MOV AH, 3CH             ; create the file

        INT 21H

        PUSH AX                 ; push file handle onto stack.

        LEA DX, MSG28           ; ask if data is to

        CALL DISP1              ; be input

        CALL READCH             ; read choice

        AND AL, 0FH

        CMP AL, 9               ; if choice = 'y' or 'y'

        JNZ NO

        LEA DX, MSG29

        CALL DISP1

        POP BX                  ; retrieve file handle from stack.

        MOV BUFFER1[1], 0

WRITE :         CALL READCH     ; read data character by character.

        MOV BUFFER1[0], AL

        CMP BUFFER1[0], 27      ; check if character is 'esc'(stop).

        JZ NO

        CMP BUFFER1[0], 0DH

        JNE NEOL

        LEA DX, MSG26

        CALL DISP1

        MOV SI, DX
```

```
          MOV BYTE PTR DS:[SI + 2], 0
          MOV CX, 3
          JMP COM
NEOL :MOV CX, 1
          LEA DX, BUFFER1[0]
COM :  MOV AH, 40H                 ; write to the file
          INT 21H
          MOV BYTE PTR DS:[SI + 2], '$'
          JMP WRITE
          ENDING: JMP ENDINGII
NO :    LEA DX, MSG16               ; creation successful
          CALL DISP1
          CALL READCH
          JMP BEGIN
```

DELETING A FILE

```
DELETE:
CALL SIMPLE
LEA DX, MSG3                        ; module for deleting a file
          CALL DISP1
          CALL READ1                ; read name of file to be deleted
          LEA DX, BUFFER1[2]
          MOV AH, 41H               ; delete the file
          INT 21H
          CMP AX, 2                 ; error if file not found
          JNZ ERR2
          LEA DX, MSG14
          CALL DISP1
          JMP ENDD
ERR2 :  CMP AX, 5                   ; error if access denied
```

```
          JNZ DONE
          LEA DX, MSG15
          CALL DISP1
          JMP ENDD
DONE :  LEA DX, MSG17                 ; delete successful
          CALL DISP1
ENDD :          CALL READCH
          JMP BEGIN
```

RENAME A FILE

```
RENAME:
CALL SIMPLE
LEA DX, MSG4                          ; module for renaming a file
          CALL DISP1
          CALL READ1                   ; read name of file to be renamed
          LEA DX, MSG5
          CALL DISP1
          CALL READ2                   ; read new name of file
          PUSH DS
          POP ES
          LEA DX, BUFFER1[2]
          LEA DI, BUFFER2[2]
          MOV AH, 56H                 ; rename file
          INT 21H
          CMP AX, 2                   ; error if file not found
          JNZ ERR3
          LEA DX, MSG14
          CALL DISP1
          JMP ENDE
```

```
ERR3 :  CMP AX, 3                    ; error if path not found
        JNZ ERR4
        LEA DX, MSG14
        CALL DISP1
        JMP ENDE
ERR4 :  CMP AX, 5                    ; error if access denied
        JNZ REN
        LEA DX, MSG15
        CALL DISP1
        JMP ENDE
REN :   LEA DX, MSG18                ; rename successful
        CALL DISP1
ENDE :  CALL READCH
        JMP BEGIN
```

VIEW – EDIT A FILE

```
LEA DX, MSG30                        ; module to view the
        CALL DISP1                   ; contents of a file
        CALL READ1                   ; read name of file
        LEA DX, MSG26
        CALL DISP1
        CALL DISP1
        LEA DX, BUFFER1[2]           ; open the file
        MOV AX, 3D02H
        INT 21H
        MOV BUFFER2[0], 0
        CMP AX, 2                    ; error if file not found
        JNZ V_ERR
        LEA DX, MSG14
```

```
            CALL DISP1
            JMP ENDV
V_ERR : CMP AX, 3                    ; error if path not found
            JNZ CONT2
            ;JNZ PUSH
            LEA DX, MSG21
            CALL DISP1
            JMP ENDV
CONT2 :         MOV BX, AX
            PUSH AX
            MOV CX, 1
            LEA DX, BUFFER1
            MOV AH, 3FH                  ; read the file
            INT 21H
            CMP AX, 0                    ; stop if end-of-file
        JZ PUSH
        JNZ SHOW2
ENDINGI: JMP ENDINGS
SHOW2 :         MOV BUFFER1[1], '$'
            LEA DX, BUFFER1
            CALL DISP1
            POP AX
            JMP CONT2
            CMP BUFFER1[0], 0DH
      ; JNZ SHOW
            INC BUFFER2[0]
            CMP BUFFER2[0], 23           ; check if end of page
            JNZ SHOW1TEMP
SHOW1TEMP:          JMP SHOW1
            ;JNZ PUSH
            LEA DX, MSG27
            CALL DISP1
            CALL READCH
            MOV BUFFER2[0], 0
            LEA DX, MSG26
            CALL DISP1
PUSH:   MOV AH,02
        MOV AL,0
            MOV CX,0
            MOV DX,10
            INT 21
            ;INT 21H
            ;PUSH AX                  ; push file handle onto stack.
            LEA DX, MSG28             ; ask if data is to
            CALL DISP1                ; be input
            CALL READCH               ; read choice
            AND AL, 0FH
            CMP AL, 9                 ; if choice = 'y' or 'y'
            JNZ NO1
            LEA DX, MSG29
            CALL DISP1
            POP BX                   ; retrieve file handle from stack.
            MOV BUFFER1[1], 0
WRITE1 :        CALL READCH           ; read data character by character.
            MOV BUFFER1[0], AL
            CMP BUFFER1[0], 27       ; check if character is 'esc'(stop).
            JZ NO1
            CMP BUFFER1[0], 0DH
            JNE NEOL1
            LEA DX, MSG26
            CALL DISP1
            MOV SI, DX
            MOV BYTE PTR DS:[SI + 2], 0
            MOV CX, 3
            JMP COM1
NEOL1 :         MOV CX, 1
            LEA DX, BUFFER1[0]
COM1: MOV AH, 40H                   ; write to the file
            INT 21H
            MOV BYTE PTR DS:[SI + 2], '$'
            JMP WRITE1
NO1 :   LEA DX, MSG16               ; creation successful
            CALL DISP1
            CALL READCH
            JMP BEGIN
            ;JMP CR_FILE
SHOW1 :         MOV BUFFER1[1], '$'
            LEA DX, BUFFER1
            CALL DISP1
            POP AX
            JMP CONT2
ENDV :  CALL READCH
            JMP BEGIN
```

COPY A FILE

```
COPY:
CALL SIMPLE
LEA DX, MSG6                        ; module for copying a file
            CALL DISP1              ; read name of file to
            CALL READ1              ; to be copied
```

```
          MOV CX, BX
          LEA DX, MSG7
          CALL DISP1
          CALL READ2                    ; read path of destination
          MOV BUFFER2[BX], '\'          ; directory
          INC BX
          MOV AX, 2
CP :    MOV SI, AX                      ; concatenating path and filename
          MOV DL, BUFFER1[SI]
          MOV BUFFER2[BX], DL
          INC BX
          INC AL
          CMP CX, AX
          JNE CP
          MOV CX, 0
          LEA DX, BUFFER2[2]            ; create the file in
          MOV AH, 3CH                   ; destination directory
          INT 21H
          CMP AX, 3                     ; display error message
          JNZ CONT                      ; if path not found
          LEA DX, MSG21
          CALL DISP1
          JMP COPY                      ; on error read data again
CONT :PUSH AX
          LEA DX, BUFFER1[2]            ; open source file
          MOV AX, 3D00H
          INT 21H
          PUSH AX
RD :      POP BX
          LEA DX, BUFFER1
          MOV CX, 80H             ; read source file
          MOV AH, 3FH
          INT 21H
          CMP AX, 0               ; check if entire file
          JZ FIN                  ; has been read
          MOV CX, BX
          POP BX
          PUSH BX
          PUSH CX
          MOV CX, AX
          LEA DX, BUFFER1         ; write into new file to
          MOV AH, 40H             ; complete copy task
          INT 21H
          JMP RD                  ; read file further
FIN :     LEA DX, MSG25           ; copy successful
          CALL DISP1
          CALL READCH
          JMP BEGIN
```

CREATING A DIRECTORY

```
CRDIR :
CALL SIMPLE
LEA DX, MSG8                      ; module for creating
          CALL DISP1              ; a directory
          CALL READ1              ; read name of directory
          LEA DX, BUFFER1[2]      ; to be created
          MOV AH, 39H             ; create directory
          INT 21H
          CMP AX, 3               ; error if path not found
```

```
            JNZ ERR5
            LEA DX, MSG21
            CALL DISP1
            JMP ENDF
ERR5 :  CMP AX, 5                     ; error if access denied
            JNZ DONE1
            LEA DX, MSG15
            CALL DISP1
            JMP ENDF
DONE1 : LEA DX, MSG19                 ; creation successful
            CALL DISP1
ENDF :CALL READCH
            JMP BEGIN
```

REMOVING A DIRECTORY

```
REDIR :
CALL SIMPLE
LEA DX, MSG9                          ; module for removing directory
            CALL DISP1
            CALL READ1                ; read name of directory to
            LEA DX, BUFFER1[2]        ; be removed
            MOV AH, 3AH               ; remove directory
            INT 21H
            CMP AX, 3                 ; error if path not found
            JNZ ERR6
            LEA DX, MSG21
            CALL DISP1
            JMP ENDG
ERR6 :  CMP AX, 5                     ; error if access denied
```

```
          JNZ DONE2
          LEA DX, MSG15
          CALL DISP1
          JMP ENDG
DONE2 : LEA DX, MSG20             ; deletion successful
          CALL DISP1
ENDG :    CALL READCH
      JMP BEGIN
```

CHANGING A DIRECTORY

```
CHDIR :
CALL SIMPLE
LEA DX, MSG10                     ; module for changing directory
          CALL DISP1
          CALL READ1              ; read name of directory to
          LEA DX, BUFFER1[2]      ; be changed to
          MOV AH, 3BH             ; change directory
          INT 21H
          CMP AX, 3               ; error if path not found
          JNZ DONE3
          LEA DX, MSG21
          CALL DISP1
          JMP ENDH
DONE3 : LEA DX, MSG22             ; change successful
          CALL DISP1
ENDH :    CALL READCH
          JMP BEGIN

CH_DRV :LEA DX, MSG13             ; module for changing drive
```

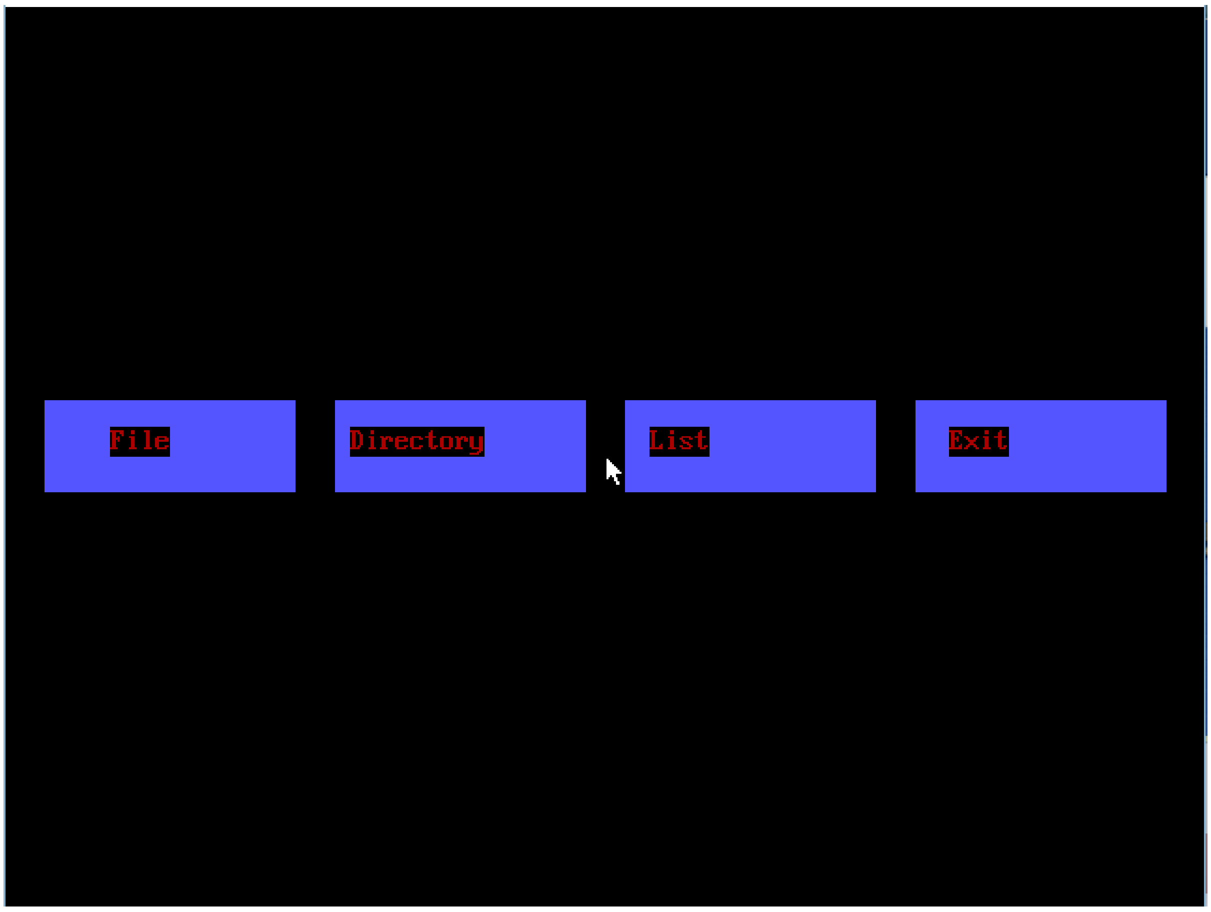
```
          CALL DISP1
          CALL READ1                  ; read name of drive
          MOV DL, 0
          CMP BUFFER1[2], 'A'
          JZ FLOPPY
          CMP BUFFER1[2], 'C'
          MOV DL, 2
FLOPPY :MOV AH, 0EH                   ; change drive
          INT 21H
          LEA DX, MSG23               ; change successful
          CALL DISP1
          CALL READCH
          JMP BEGIN
```

LISTING A DIRECTORY

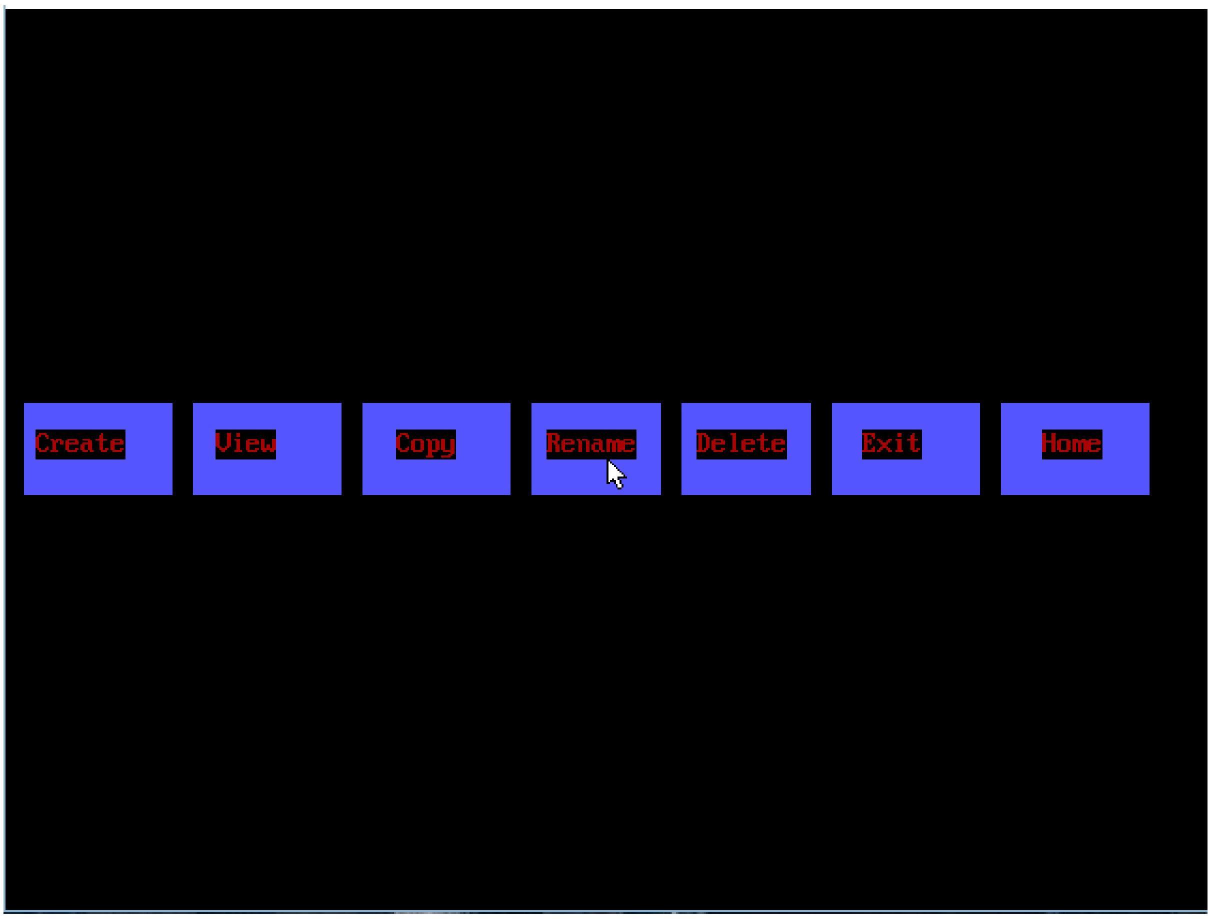
```
LISTING:
CALL SIMPLE
          MOV AX, 3H                  ; module for displaying
          INT 10H                     ; contents of directory
          LEA DX, MSG24
          MOV CX, 0
          MOV AH, 4EH                 ; get first file
          INT 21H                     ; in directory
          CMP AX, 18                  ; check if no files
          JNZ LIST                    ; in directory
          LEA DX, MSG14               ; display message
          CALL DISP1                  ; 'file not found'
          CALL READCH
          JMP BEGIN
```

```
LIST :  MOV AH, 2FH                  ; get dta address
        INT 21H
        MOV BYTE PTR ES:[BX + 42], 0
        ADD BX, 1EH
        MOV BUFFER1[0], 0
CHAR :MOV DL, BYTE PTR ES:[BX]  ; get character of
        INC BX                        ; filename from dta
        INC BUFFER1[0]
        CMP DL, '.'                  ; check if extension
        JNZ CONT3                    ; is starting
CONT4 :         LEA DX, MSG31
        CALL DISP1
        INC BUFFER1[0]
        CMP BUFFER1[0], 0BH          ; check for end of filename
        JNE CONT4                    ; buffer - 13 characters
        JMP CHAR
CONT3 :         MOV AH, 02H           ; display character
        INT 21H                      ; of filename
        CMP DL, 0                    ; check for end
        JNE CHAR                     ; of file name
        LEA DX, MSG26
        CALL DISP1
        INC CX
        CMP CX, 23                   ; check for end of page
        JNE CONT1
        LEA DX, MSG27
        CALL DISP1
        CALL READCH
        MOV CX, 0
        LEA DX, MSG26
        CALL DISP1
CONT1 :     MOV AH, 4FH          ; get next file
        INT 21H
        JNC LIST
        LEA DX, MSG27
        CALL DISP1
        CALL READCH
        JMP BEGIN1
```

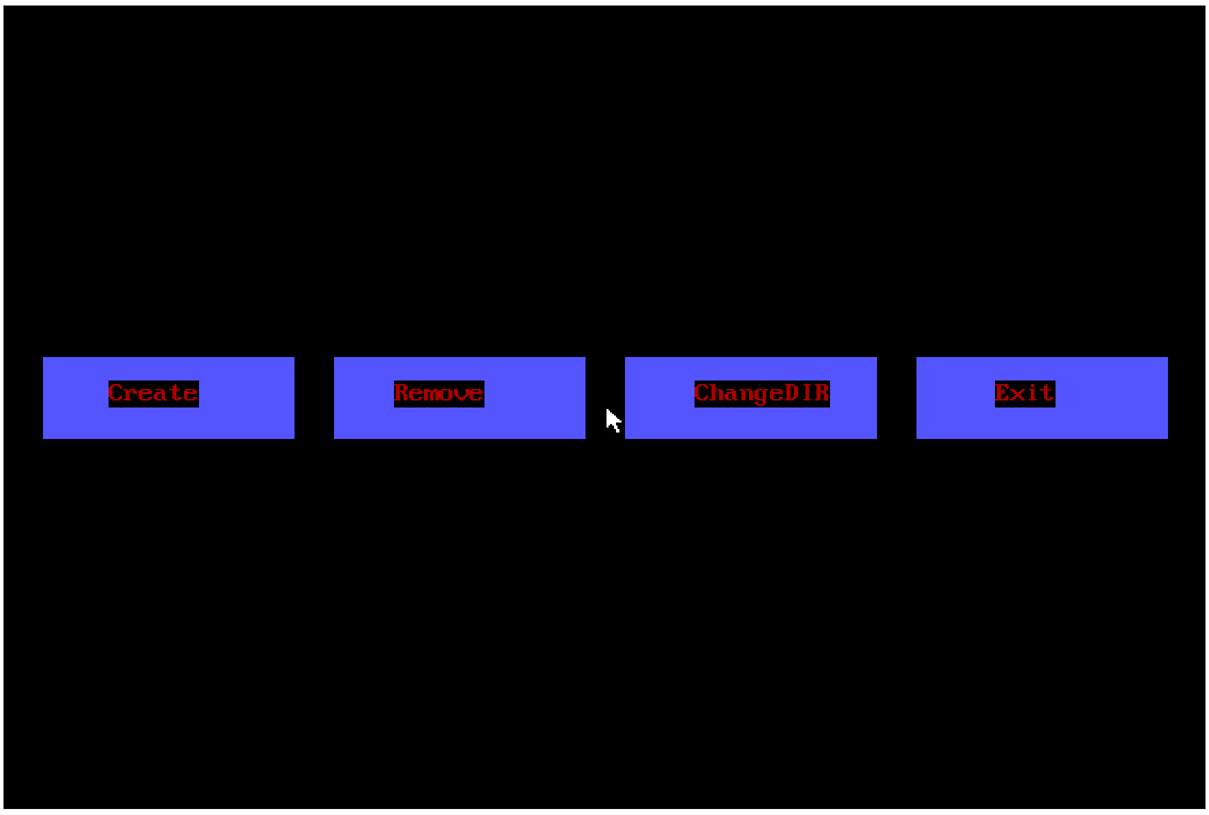
SCREENSHOTS



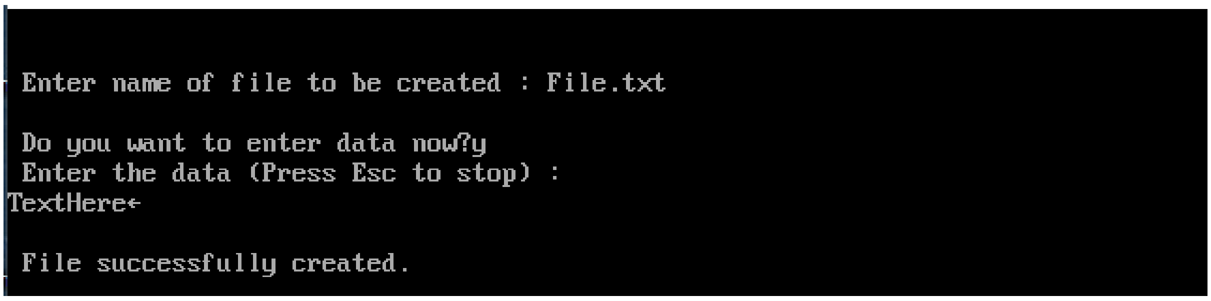
Screenshot 1 Homepage



Screenshot 2 FileButtonClicked



Screenshot 3 DirectoryButtonClicked

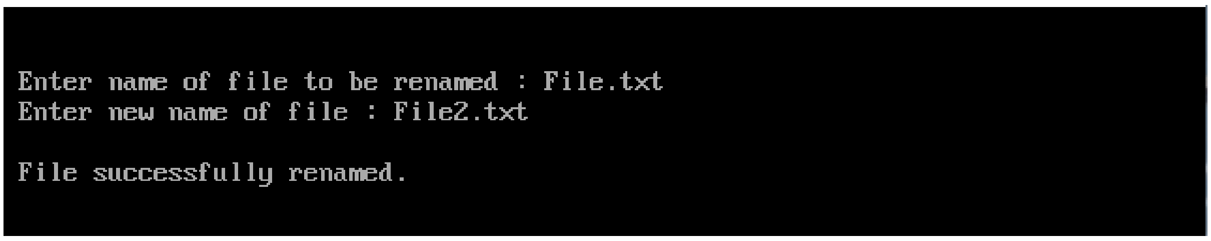


```
 Enter name of file to be created : File.txt

 Do you want to enter data now?y
 Enter the data (Press Esc to stop) :
TextHere←

 File successfully created.
```

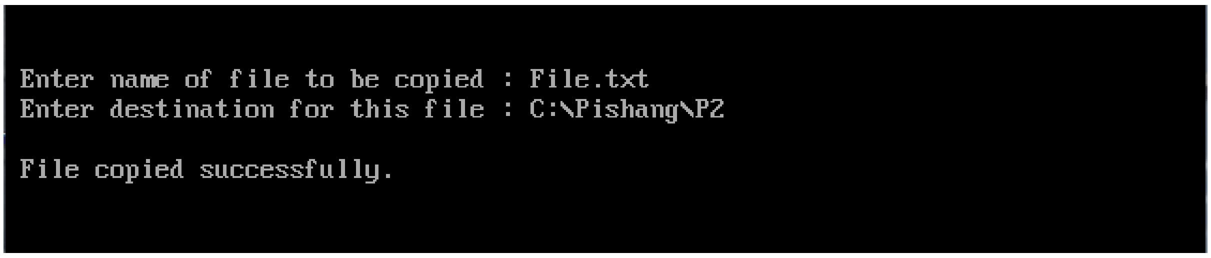
Screenshot 4 CreateFile



```
Enter name of file to be renamed : File.txt
Enter new name of file : File2.txt

File successfully renamed.
```

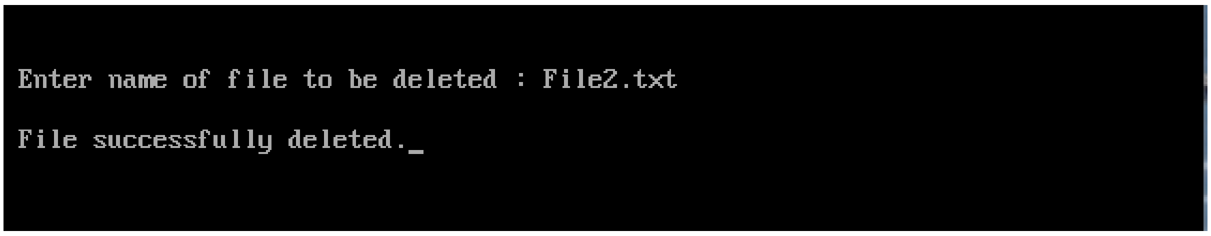
Screenshot 5 RenameFile



```
Enter name of file to be copied : File.txt
Enter destination for this file : C:\Pishang\P2

File copied successfully.
```

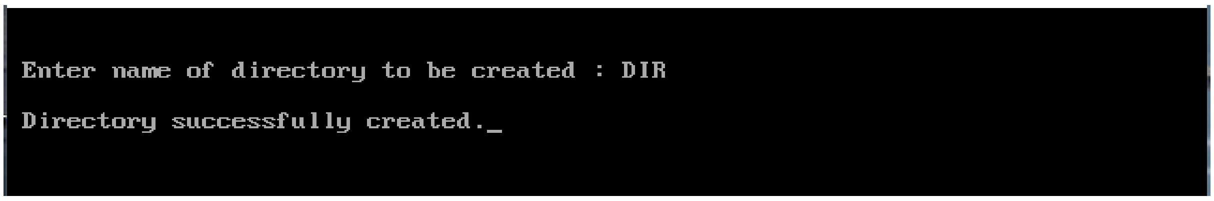
Screenshot 6 CopyFile



```
Enter name of file to be deleted : File2.txt

File successfully deleted._
```

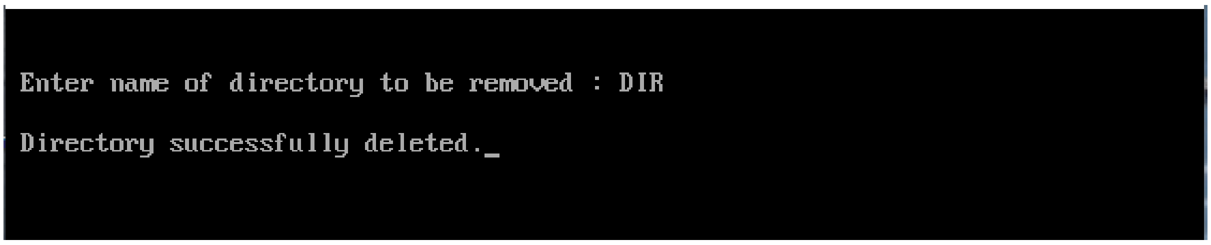
Screenshot 7 DeleteFile



```
 Enter name of directory to be created : DIR

 Directory successfully created._
```

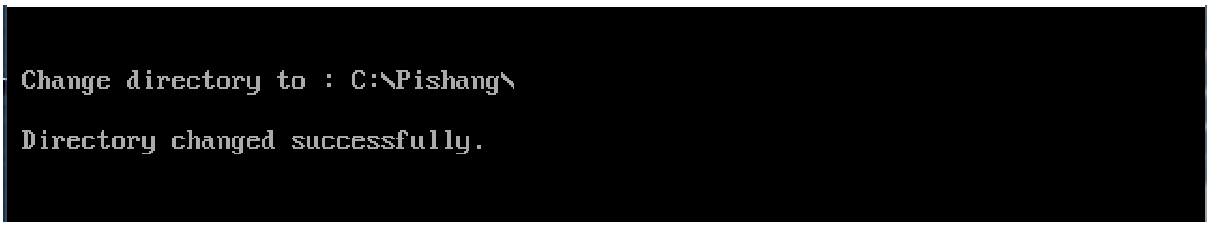
Screenshot 8 CreateDirectory



```
Enter name of directory to be removed : DIR

Directory successfully deleted._
```

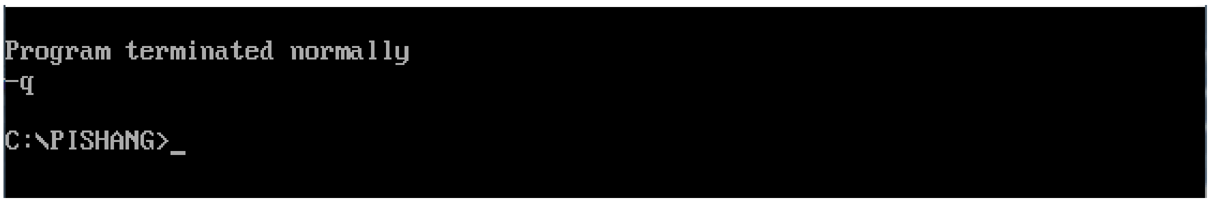
Screenshot 9 RemoveDirectory



```
Change directory to : C:\Pishang\

Directory changed successfully.
```

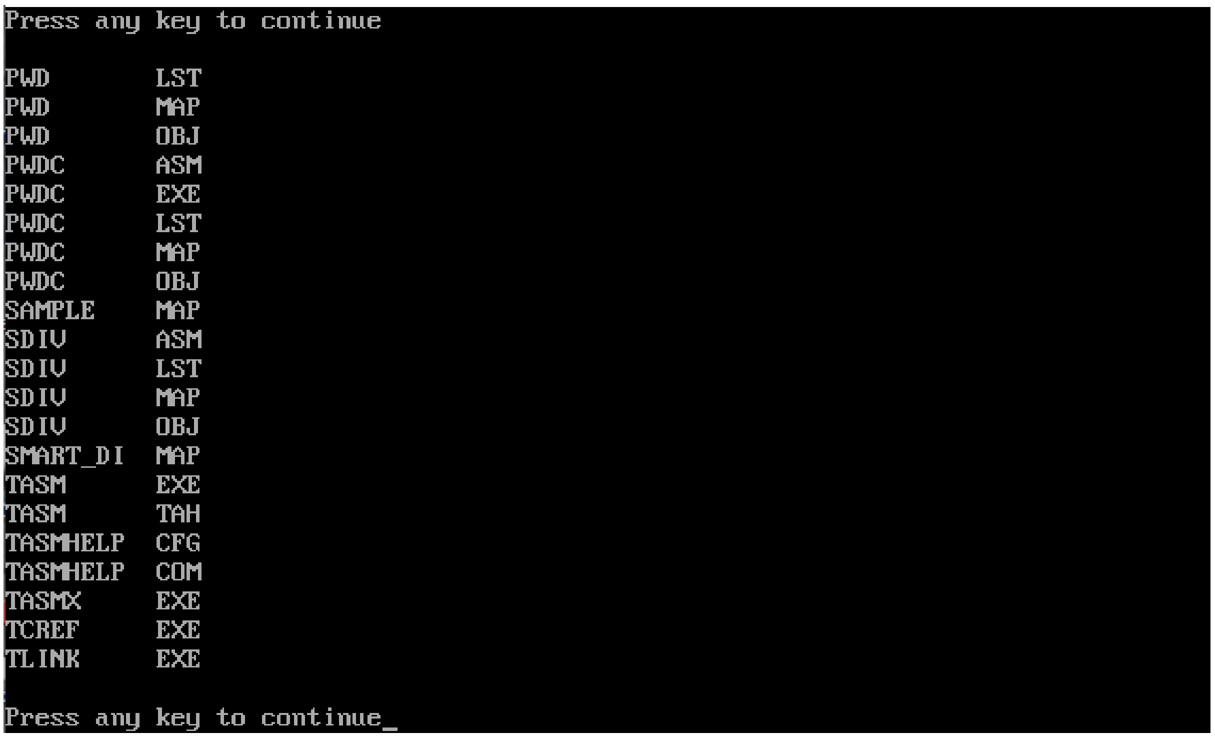
Screenshot 10 ChangeDirectory



```
Program terminated normally
-q

C:\PISHANG>_
```

Screenshot 11 DirectoryChanged-at-prompt-checked-after-exiting-program



```
Press any key to continue

PWD       LST
PWD       MAP
PWD       OBJ
PWDC      ASM
PWDC      EXE
PWDC      LST
PWDC      MAP
PWDC      OBJ
SAMPLE    MAP
SDIV      ASM
SDIV      LST
SDIV      MAP
SDIV      OBJ
SMART_DI  MAP
TASM      EXE
TASM      TAH
TASMHELP  CFG
TASMHELP  COM
TASMX     EXE
TCREF     EXE
TLINK     EXE

Press any key to continue_
```

Screenshot 12 ListingDirectory