Lab 6 : LÀM VIỆC VỚI TẬP TIN

1. **Chuẩn đầu ra :** Sau bài này, người học có thể :
   - ✓ Sử dụng các system call để xử lý tập tin
2. **Chuẩn bị :** Đọc trước phần lý thuyết về system call.
3. **Phương tiện :**
   - ✓ Máy vi tính.
   - ✓ Chương trình nasm.
4. **Thời lượng : 4 tiết**
5. **Tóm tắt lý thuyết**

5.1. Creat file

- EAX ←8
- EBX ← pointer to ASCIIZ pathname
- ECX ← file permission
- INT 80h
- Return : EAX ← interger file descriptor

5.2. Open file

- EAX ←5
- EBX ← pointer to ASCIIZ pathname
- ECX ← file access mode (0x00 = readonly, 0x01 = write only, 0x02 = read/write)
- EDX ← file permissions
- INT 80h
- Return : EAX ← interger file descriptor

5.3. Write to file

- EAX ←4
- EBX ← file descriptor
- ECX ← pointer to output buffer
- EDX ← number of bytes to write
- INT 80h
- Return : EAX ← number of bytes actually written
- Notes : write to creen using stdout descriptor =1

5.4. Read from file

- EAX ←3
- EBX ← file descriptor
- ECX ← pointer to input buffer
- EDX ← number of bytes to read
- INT 80h
- Return : EAX ← number of bytes actually to read
- Notes : read from keyboard using stdin descriptor =0

5.5. Close file

- EAX ←6
- EBX ← file descriptor
- INT 80h
-

6. **Nội dung thực hành**

6.1. Nạp chương trình sau vào

```
demonstrates file open, file read (in hunks of 8192) and file write
(the whole file!)This program reads and prints itself:
; syscall1_64.asm   demonstrate system, kernel, calls
; Compile:   nasm -f elf64 syscall1_64.asm
; Link       gcc -o syscall1_64 syscall1_64.o
; Run:       ./syscall1_64
;

section     .data
msg:  db  "syscall1_64.asm running",10; the string to print,
10=crlf
len:  equ $-msg  ; "$" means here, len is a value, not an address
msg2: db  "syscall1_64.asm finished",10
len2: equ $-msg2
msg3: db  "syscall1_64.asm opened",10
len3: equ $-msg3
msg4: db  "syscall1_64.asm read",10
len4: equ $-msg4
msg5: db  "syscall1_64.asm open fail",10
len5: equ $-msg5
msg6: db  "syscall1_64.asm another open fail",10
len6: equ $-msg6
msg7: db  "syscall1_64.asm read fail",10
len7: equ $-msg7

name: db  "syscall1_64.asm",0 ; "C" string also used by OS
fd:   dq  0              ; file descriptor
flags:dq  0              ; hopefully read-only
      section .bss
line: resb  8193         ; read/write buffer 16 sectors of 512
lenbuf:     resq     1   ; number of bytes read

      extern  open
      global main
      section .text
main:
       push    rbp       ; set up stack frame

; header msg
      mov   rdx,len      ; arg3, length of string to print
      mov   rcx,msg      ; arg2, pointer to string
      mov   rbx,1        ; arg1, where to write, screen
      mov   rax,4        ; write command to int 80 hex
      int   0x80         ; interrupt 80 hex, call kernel
```

```
open1:
      mov   rdx,0        ; mode
      mov   rcx,0        ; flags, 'r' equivalent O_RDONLY
      mov   rbx,name     ; file name to open
      mov   rax,5        ; open command to int 80 hex
      int   0x80         ; interrupt 80 hex, call kernel
        mov [fd],rax     ; save fd
      cmp   rax,2        ; test for fail
      jg    read         ; file open

; file open failed msg5
      mov   rdx,len5     ; arg3, length of string to print
      mov   rcx,msg5     ; arg2, pointer to string
      mov   rbx,1        ; arg1, where to write, screen
      mov   rax,4        ; write command to int 80 hex
      int   0x80         ; interrupt 80 hex, call kernel

read:
; file opened msg3
      mov   rdx,len3     ; arg3, length of string to print
      mov   rcx,msg3     ; arg2, pointer to string
      mov   rbx,1        ; arg1, where to write, screen
      mov   rax,4        ; write command to int 80 hex
      int   0x80         ; interrupt 80 hex, call kernel

doread:
      mov   rdx,8192     ; max to read
      mov   rcx,line     ; buffer
      mov   rbx,[fd]     ; fd
      mov   rax,3        ; read command to int 80 hex
      int   0x80         ; interrupt 80 hex, call kernel
      mov   [lenbuf],rax ; number of characters read
      cmp   rax,0        ; test for fail
      jg    readok       ; some read

; read failed msg7
      mov   rdx,len7     ; arg3, length of string to print
      mov   rcx,msg7     ; arg2, pointer to string
      mov   rbx,1        ; arg1, where to write, screen
      mov   rax,4        ; write command to int 80 hex
      int   0x80         ; interrupt 80 hex, call kernel
      jmp   fail         ; nothing read

; file read msg4
readok:
      mov   rdx,len4     ; arg3, length of string to print
      mov   rcx,msg4     ; arg2, pointer to string
```

```
        mov    rbx,1        ; arg1, where to write, screen
        mov    rax,4        ; write command to int 80 hex
        int    0x80         ; interrupt 80 hex, call kernel

write:
        mov    rdx,[lenbuf] ; length of string to print
        mov    rcx,line     ; pointer to string
        mov    rbx,1        ; where to write, screen
        mov    rax,4        ; write command to int 80 hex
        int    0x80         ; interrupt 80 hex, call kernel

fail:
; finished msg2
        mov    rdx,len2     ; arg3, length of string to print
        mov    rcx,msg2     ; arg2, pointer to string
        mov    rbx,1        ; arg1, where to write, screen
        mov    rax,4        ; write command to int 80 hex
        int    0x80         ; interrupt 80 hex, call kernel

        mov    rbx,0        ; exit code, 0=normal
        mov    rax,1        ; exit command to kernel
        int    0x80         ; interrupt 80 hex, call kernel
```

- **Lưu chương trình với tên**
- **Biên dịch**
- **Liên kết**
- **Chạy thử**

6.2. Nhập chương trình sau và chạy thử

```
;;;reader.asm
;;; A simple program that says hello and
;;; then reads in a string from the STDIN
;;; It changes any upper case letters to lower case.
;;; To run:
;;; nasm -f      elf -F stabs reader.asm
;;; ld -o reader reader.o
;;; ./reader

%assign SYS_EXIT        1
%assign READ            3
%assign WRITE           4
%assign STDOUT          1
%assign STDIN           0
%assign ENDL            0x0a

;; -------------------------
;; data segment
;; -------------------------
```

```
section   .data
msg db    "hello"
    db      ENDL
MSGLEN  equ     6
strEnd:   db '\0'   ;; 0x00

section .bss
inmsg:    resb  255
inlen:    resd  1

;; -------------------------
;; code area
;; -------------------------

section   .text
    global _start

_start:
    mov     eax,WRITE         ;4
    mov     ebx,STDOUT        ;1
    lea     ecx,[msg]         ;address of source
    mov     edx,MSGLEN        ; length (num of characters)

    int    0x80
    mov   edi, 1
   mov byte [inmsg], 0 ;init with end of string for the loop

nxtchr:   cmp   byte[inmsg + edi - 1], ENDL
    je    endlp
    mov   eax, READ ; 3  place READ value in eax instead of WRITE
    mov   ebx, STDIN   ; 0 place STDIN value in ebx instead of
STDOUT
    lea   ecx, [inmsg + edi] ; address of destination
    mov   edx, 1          ; length (num of char)
    int   0x80
    inc   edi
    jmp   nxtchr

    mov   ecx, -1
loopit:   inc   ecx
    mov   al, [inmsg + ecx]
    cmp   al, [strEnd]
    JE    endlp
    cmp   al, 0x41
    JB    loopit
    cmp   al, 0x5A
    JAE   loopit
```

```
    add    al, 0x20
    mov    [inmsg + ecx], al
    jmp    loopit

endlp:    mov    eax, WRITE   ; write the input string back out
    mov    ebx, STDOUT
    lea    ecx, [inmsg]
    mov    edx, edi     ; edi now contains the string length
    int    0x80

    ;; exit()

    mov    eax,SYS_EXIT
    mov    ebx,0
    int    0x80         ; final system call
```

6.3.