

```

; =====
; tdtrois.asm
;
; Écrire un programme qui lit dix nombres (entiers non signés)
; au clavier et écrit dans le fichier résultat ces mêmes nombres
; triés au préalable par ordre croissant.
;
; Auteur:      Pbt
; Date:       avril 2008
; Build:      nasm tdtrois.asm -f elf
;             gcc tdtrois.o -o tdtrois
; =====
BITS 32
GLOBAL main

; =====
; Déclaration des constantes
N          EQU      10
T_BUFFER   EQU      10

; =====
; Déclaration des variables
SECTION .data
filename   DB        "resultats.data",0

; =====
; Déclaration des variables non-initialisées
SECTION .bss
naturels   resd      N          ; tableau de nombres
inb        resb      T_BUFFER+1 ; buffer, 2^32 ~= 4294967296
outb       resb      T_BUFFER   ; buffer
fd         resd      1          ; file descriptor

; =====
; Code, fonction principale main
SECTION .code

;
; -----main-----
;
main:
; Lecture des N nombres
mov     ECX,N
mov     ESI,0
.repeter:
    push ECX
    call lirenombre
    mov [naturels+ESI],EAX
    add ESI,4
    pop ECX
    loop .repeter

; Tri du tableau
push dword N
push dword naturels
call tri
add ESP,4*2

; Ouverture du fichier
; int open(const char *pathname, int flags);
mov EAX,5
mov EBX,filename
mov ECX,000001000010b
mov EDX,100100100b
int 0x80
cmp EAX,-1
je .exit
mov [fd],EAX

```

```

; Écriture du fichier
mov ECX,N
mov ESI,0
.repeter:
    push ECX
    mov EAX,[naturels+ESI]
    call ecrireNombres
    add ESI,4
    pop ECX
    loop .repeter

; Fermeture du fichier
; int close(int fd); (6)
mov EAX,6
mov EBX,[fd]
int 0x80

.exit:
ret
; end main

;
; -----lirenombre-----
; Permet de lire un nombre de maximum 10 (T_BUFFER) chiffres au
; clavier (éviter de lire les nombres de 9 chiffres ^^).
; Le nombre lu est un entier positif.
; @return:      EAX - le nombre lu
;
lirenombre:
    push EBX
    push ECX
    push EDX
    push ESI
    ; ssize_t read(int fd, void *buf, size_t count); (3)
    mov EAX,3
    mov EBX,0
    mov ECX,inb
    mov EDX,T_BUFFER
    int 0x80
    mov ECX,EAX
    ; Ne pas lire le caractère de fin si nlu < T_BUFFER
    ; read retourne le nombre de caractères lus+1 si nlus < count,
    ; sinon retourne count.
    ; Ceci pose u problème lorsque l'on lit T_BUFFER-1 caractères;
    ; read retourne T_BUFFER hors que l'on a lu que T_BUFFER-1 char.
    .si
    cmp ECX,T_BUFFER
    je .finsi
    ; Ne pas lire le caractère de fin lorsque le buffer n'est pas plein
    dec ECX
    .finsi
    ; Lecture premier char
    ; Décompte de ce char dans ECX
    ; Conversion du char en "quantité" par soustraction de '0'
    mov ESI,0
    mov DL,[inb+ESI]
    inc ESI
    dec ECX
    sub DL,'0'
    movsx EAX,DL
    mov EBX,10
    ; Tant qu'il reste des caractères dans le buffer, les ajouter à EAX
    ; après multiplication de EAX par 10.

```

```

    .repeter:
        cmp ECX,0
        jle .finrepeter
        push ECX
        mov EDX,0
        mul EBX
        mov DL,[inb+ESI]
        inc ESI
        sub DL,'0'
        movsx EDX,DL
        add EAX,EDX
        pop ECX
        dec ECX
        jmp .repeter
    .finrepeter
    pop ESI
    pop EDX
    pop ECX
    pop EBX
ret
; end lirenombre

;
; -----tri-----
; Trie le tableau de nombres passé en paramètres.
; Utilisation d'un tri par insertion.
; C'est un tableau de "double"
; @param pile 1 - nombre d'éléments (EBP+12)
; @param pile 2 - adresse du tableau (EBP+8)
tri:
    push EBP
    mov EBP,ESP
    pusha
    ; EBX, Adresse du tableau
    ; ECX,Nombre d'éléments
    mov EBX,[EBP+8]
    mov ECX,[EBP+12]
    ; Je commence à l'elt d'indice 1
    mov ESI,4
    .repeter:
        push ECX
        ; EDX, elt en cours
        ; ESI, indice de boucle de l'elt à insérer
        ; EDI, indice de boucle de recherche de l'endroit d'insertion
        mov EDX,[EBX+ESI]
        mov EDI,0
        .while:
            cmp EDX,[EBX+EDI]
            jle .endwhile
            add EDI,4
            jmp .while
        .endwhile:
        ; Je décale les elts vers l'avant de EDI à ESI
        push ESI
        .repeterr:
            mov EAX,[EBX+ESI-4]
            mov [EBX+ESI],EAX
            sub ESI,4
            cmp ESI,EDI
            jge .repeterr
        pop ESI
        ; Insertion à la place vacante
        mov [EBX+EDI],EDX
        ; Passe à l'elt suivant
        add ESI,4
        pop ECX
        dec ECX
        cmp ECX,1
        jg .repeter
    popa
    pop EBP
ret
; end tri

```

```
;
; -----écrireNombres-----
; Convertit le nombre et l'écrit dans le fichier pointé par fd.
; On suppose que le fichier est ouvert en écriture.
;
; @param EAX - le nombre à afficher
; @use inb - inb qui contiendra la chaîne à afficher
; @use fd - file descriptor, le descripteur du fichier
;
ecrireNombres:
    pusha
    mov EBX,10
    ; ESI pointe vers la fin du buffer
    mov ESI,outb
    add ESI,T_BUFFER-1
    ; J'écris d'abord un passage à la ligne
    mov byte [ESI],10
    dec ESI
    mov EDI,1 ; nombre de char à écrire
    .repeter
        mov EDX,0
        div EBX
        add EDX,'0'
        mov [ESI],DL
        dec ESI
        inc EDI
        cmp EAX,0
        jg .repeter
    ; ssize_t write(int fd, const void *buf, size_t count);
    mov EAX,4
    mov EBX,[fd]
    inc ESI ; ESI est un char trop "à gauche"
    mov ECX,ESI
    mov EDX,EDI
    int 0x80
    popa
ret
; end écrireNombres

;END
```