Université de Tunis El-Manar Faculté des Sciences de Tunis Département des Sciences de l’Informatique

IF3

**Mini Projet Assembleur N3**

*Projet développer par : Aymen Ben Mhamed(Amphi B GR03) Ayda Rekaya (Amphi A GR03)*

***Section : Ingéniorat en Informatique IF3   
Matière : Système d’Exploitation***

***TP système d’exploitation N°3 :***

***Mini Projet Assembleur***

***Manipulation des fichiers***

***Sommaire :***

|  |  |
| --- | --- |
| **Page 1** | ***Sommaire*** |
| **Page 3** | * ***Introduction à l’assembleur*** * ***Introduction à l’utilisation du Macros*** * ***Introduction aux Fichiers en assembleur*** |
| **Page 8** | ***Problématique*** |
| **Page 9** | ***Programme assembleur***  ***Code source (TP3 :Manipulation des fichiers)*** |
| **Page 19** | ***Étape de passage du code source en code exécutable*** |
| **Page 21** | ***Conclusion*** |

***Section : Ingéniorat en Informatique IF3   
Matière : Système d’Exploitation***

***TP système d’exploitation N°3:***

***Mini Projet Assembleur***

***Manipulation des fichiers***

Introduction à l’assembleur :

*Contrairement aux langages évolués, l'assembleur, ou « langage d’assemblage » est constitué d'instructions directement compréhensibles par le microprocesseur : c'est ce qu'on appelle un langage de bas niveau. Il est donc intimement lié au fonctionnement de la machine. C'est pourquoi il est relativement difficile à assimiler, en tout cas beaucoup plus que les langages de haut niveau.*

*Cela explique également pourquoi il existe au moins autant de langages d’assemblage que de modèles de microprocesseurs*

Introduction à l’utilisation des Macros :

*Un programme sera souvent constitué de plusieurs sous-programmes. Cela facilite la lisibilité et la clarté du code.*

*Un macro (ou sous routine ou en encore sous programme) permet de cataloguer un algorithme (Procédure ou fonction ou macro) pour pouvoir ensuite faire appel à cet algorithme à tout moment (appel de procédure ou de fonction ou appel de macro).*

*Les sous-programmes sont en général prévus pour exécuter un traitement particulier.*

*En règle générale, les sous-programmes constituent une excellente méthode de découpage de gros programme en petites parties beaucoup plus faciles à manipuler, il ne faut donc pas hésiter à les utiliser.*

LES FICHIERS :

LES INTERRUPTIONS DU DOS RELATIVES AUX FICHIERS:*Remarque préliminaire :  
 Cette partie présente quelques unes des interruptions du DOS qui servent à manier les fichiers d'un disque.*I. LECTURE ET ECRITURE DE FICHIERS AVEC LES HANDLES :*Sous DOS, il existe deux méthodes pour accéder aux fichiers et pour chacune d'elles un lot d'interruptions spécifiques. La première est la méthode des FCB (« File contol block »). Elle est rarement utilisée,  
aussi ne l'aborderons nous pas. Nous étudierons le principe de la  
méthode des « handles ».  
  
Pour lire ou écrire des données dans un fichier, il est nécessaire de l'ouvrir, c'est-à-dire de le charger en mémoire. Quand toutes les opérations de lecture et d'écriture auront été effectuées, le fichier devra être refermé afin d'enregistrer les éventuelles dernières modifications et surtout de libérer la mémoire occupée.* **1. OUVERTURE D'UN FICHIER :** *On ouvre un fichier en appelant la fonction 3dh de l'interruption 21h. Celle-ci attend comme paramètre dans DS:DX l'adresse d'une chaîne de caractères qui contient le chemin d'accès au fichier sur un disque, par exemple “C:\MonDoss\MonFic.txt”.  
  
Remarque :  
 il n'est pas indispensable de mentionner le chemin d'accès complet : par défaut, le fichier sera cherché à partir du dossier courant.  
  
Remarque importante :  
 La chaîne doit être impérativement suivie de l'octet 00h qui sert à marquer sa fin.  
  
Il nous faut également spécifier le mode d'accès en écrivant dans AL un 0 (si on veut ouvrir le fichier en lecture seule), un 1 (si on veut l'ouvrir en écriture seule) ou un 2 (lecture ET écriture).  
Si l'interruption échoue, le flag CF sera mis à 1 sans que le fichier soit ouvert. Dans le cas contraire, CF est mis 0 et le registre AX contient un petit nombre entier (par exemple 5) appelé « handle » (ce  
qui signifie « poignée ») du fichier. Ce handle représente le fichier.  
C'est lui qu'il faudra désormais invoquer pour effectuer des opérations de lecture ou d'écriture, et non pas le chemin d'accès.  
En effet, les chemins d'accès ne sont donc plus d'aucune utilité puisque les fichiers sont ouverts dans la mémoire vive.* **2. LECTURE DANS UN FICHIER :** *Une fois le fichier ouvert, on peut le lire avec la fonction 3fh. Il suffit de mentionner le handle dans BX, le nombre d'octets à lire dans CX, et l'adresse d'un buffer dans DS:DX.  
Au cas où vous ne sauriez pas ce qu'est un buffer (ou tampon), sachez que c'est simplement une variable (généralement une chaîne de caractères) destinée à recevoir des données (ou à en fournir). Dans  
notre cas, le buffer va recevoir les octets lus dans le fichier.  
  
Après l'appel, AX contient le nombre d'octets qui ont été effectivement lus (il peut être inférieur à la taille demandée si le fichier n'est pas assez long). En cas de problème, CF sera mis à 1.*

**3. ECRITURE DANS UN FICHIER :**

*Pour écrire des données, on procède de même avec la fonction 40h. Les paramètres sont les mêmes que pour la fonction 3fh. Le buffer contient cette fois les octets à écrire. Après l'appel, le  
nombre d'octets qui ont été effectivement écrits est stocké dans AX (il sera être inférieur à la taille spécifiée si le disque est plein).  
  
Les données sont écrites sur le disque durs que le tampon (dans la mémoire vive) est plein.***4. EXISTANCE D'UN POINTEUR DANS LE FICHIERS :**

*Une question se pose cependant :  
 À quel endroit du fichier les données sont-elles lues (ou écrites) ?*

*Réponse :  
 Quand un fichier est ouvert, un pointeur spécial pointant vers le début du fichier est créé. La première opération de lecture (ou d'écriture) se fera donc au début du fichier.  
Mais entre chaque opération, le pointeur est incrémenté de la taille des données que l'on a lues (ou écrites). La deuxième opération se fera donc sur les octets qui suivent ceux de la première.*

*Remarque :  
Il est possible de modifier directement le pointeur de fichier : voyez pour cela la fonction 42h...*

**5.FERMETURE D'UN FICHIER :**

*Pour terminer, le fichier doit être refermé. Les modifications éventuellement apportées et non enregistrées seront écrites sur le disque, et le handle sera libéré. C'est la fonction 3eh qui se charge de tout cela. Elle attend simplement le handle du fichier dans BX. Et comme d'habitude, CF vaut 1 après l'appel si deS erreurs ont été rencontées.*

*Remarque :  
 Attention lorsque vous laissez le fichier ouvert longtemps afin d'y ajouter progressivement des données ! Si le système plante, vous perdrez les données qui se trouvent dans le tampon en ce moment.  
C'est pourquoi il est conseillé de forcer régulièrement l'écriture sur le disque en refermant le fichier.*

**6. Conclusion***Le tableau suivant récapitule ces différentes étapes :*

|  |  |  |
| --- | --- | --- |
| *Fonction* | *Description* | *Paramètres* |
| *3dh* | *Ouvrir le fichier* | *- DS:DX : adresse d'une chaîne contenant le chemin d'accès.   - AL : mode d'accès* |
| *3eh* | *Fermer le fichier* | *- BX : handle* |
| *3fh* | *Lire le fichier* | *- BX : handle*  *- CX : nombre d'octets  - DS:DX : adresse d'un buffer* |
| *40h* | *Ecrire dans le fichier* | *- BX : handle*  *- CX : nombre d'octets  - DS:DX : adresse d'un buffer* |

**II. LES FONCTIONS DE RECHERCHE DE FICHIERS** *Pour rechercher un fichier (ou un dossier), on se sert des fonctions 4eh (« Find First ») et 4fh (« Find Next »).  
  
Imaginons par exemple que nous voulions chercher dans le dossier courant tous les fichiers qui portent l'extension “.com” afin de les supprimer. Comment devons-nous nous y prendre ?* **1. La fonction 4eh:**

*La fonction 4eh sert à définir des critères de recherche et à trouver le premier fichier qui*

*correspond à ces critères (s'ilexiste).  
  
On doit passer dans DS:DX l'adresse de la chaîne de caractères qui contient le masque de recherche (dans notre exemple, ce masque est“\*.com”). Par défaut, les fichiers sont cherchés dans le dossier  
courant. Mais on peut évidemment spécifier un autre chemin dans le  
masque.*

**Remarque importante :**

*Afin que le DOS puisse connaître sa taille, le masque doit impérativement être terminé par l'octet 00h !  
On écrit également dans CX les attributs des fichiers que l'on désire trouver. Si CX vaut 0, seuls les fichiers « normaux » pourront être trouvés. En fait, chaque bit de CL représente un attribut, comme le  
montre le tableau ci-dessous :*

|  |  |
| --- | --- |
| *Bit* | *Signification* |
| *1* | *Lecture seule* |
| *2* | *Fichier caché* |
| *3* | *Fichier système* |
| *4* | *Volume* |
| *5* | *Répertoire* |
| *6* | *Fichier* |
| *7* | *(Aucune...)* |
| *8* | *(Aucune...)* |

*Pour demander à la fonction 4eh de ne pas oublier les fichiers cachés, il suffit donc de charger CX*

*avec la valeur 2 (bit numéro 2 = 1). De même, l'attribut 00000111b (soit 7) nous permettra de trouver les  
fichiers en lecture seule, les fichiers cachés et les fichiers  
systèmes.  
  
Remarque :  
Ne vous souciez pas trop des bits numéro 4 et 6. Laissez-les à 0.  
  
Une fois que les paramètres ont été ajustés, on peut appeler la fonction 4eh. Si aucun fichier n'a été trouvé, le flag CF est mis à 1.  
On doit donc fait un test sur CF pour savoir si la recherche peut continuer ou si elle doit s'arrêter.  
Si au contraire la fonction a trouvé un fichier, les caractéristiques de ce fichier (i.e. son nom, sa taille, ses attributs,...) sont inscrits dans une zone de la mémoire appelée DTA (« Disk Transfer Area»).  
  
Mais où se trouve donc cette DTA et à quoi ressemble-t-elle ?  
  
Réponse :  
Par défaut, le DOS place la DTA dans le PSP de votre programme,à l'offset 80h.*

*Remarque :  
 il vous est naturellement possible de la déplacer en faisant appel à la fonction 1ah.  
  
 Voici La structure de la DTA :*

|  |  |  |
| --- | --- | --- |
| *Adresse* | *Description* | *Taille (octets)* |
| *00h* | *Lettre du lecteur (0=courant, 1=A,2=B, ...) sur lequel se trouve le fichier* | *1* |
| *01 h* | *Modèle de la recherche* | *11* |
| *0Ch* | *Réservé* | *9* |
| *15h* | *Attributs du fichier* | *1* |
| *16h* | *Heure du fichier* | *2* |
| *18h* | *Date du fichier* | *2* |
| *1Ah* | *Taille du fichier* | *4* |
| *1Eh* | *Non du fichier avec l'extension* | *13* |

Problématique :

*On se propose de developper un programme assembleur qui permet de manier les fichiers en effectuant les différentes tâches suivantes :*

* + - *Ouvrir un fichier texte dont le nom est saisi au clavier*
    - *Lire une série d'entiers codés en binaire et appartenant à l'intervalle [0,9] ;*
    - *Chaque entier saisi par ligne dans le fichier.*
    - *Lire à partir du fichier après la termination de la saisie.*
    - *Saisir un entier quelconque «****n» codé en décimal*** *et le Transformer en* ***binaire****.*
    - *Rechercher l'entier* ***n*** *saisi dans le fichier, en affichant la ligne ou se trouve l'entier* ***n***  *et signaler* ***untrouvable*** *sinon.*

Code source

Title prog\_TP3

.model small

.stack 100h

**.386**

.data

msg10 db ' Mini Projet Assembleur no 3 (Manipulation de fichiers) ',13,10,'$'

msg11 db ' developpe par ',13,10,'$'

msg12 db ' AYMEN BEN MHAMED && AYDA REKAYA ',13,10,' $'

nom\_fichier db 9,8 dup(?)

refuse DB 0ah,"Access IMPOSSIBLE$"

msg2 db "fichier inexistant$"

nomfichier db "\*Donner le nom de fichier à ouvrir $"

msg1 db "====>saisie 5 entier codé en binaire: $"

msg15 db "fichier est ouvert avec succes $"

message\_entier db 10,13,'!!chaque bit d entier en binaire doit etre soit 0 ou 1 !!$'

mesg\_err db 10,13,'\*\*veuillez entrer un entier en binaire code sur 4 bits\*\*',10,13,'$'

handle2 dw ?

entier db 9,8 dup(?)

alaligne db 13,10,'$'

entier2 db 9,8 dup(?)

somme db ?

msg8 db 13,10,' Si vous voulez sortir, appuyez sur n"importe qu"elle touche du clavier$'

msg5 db "l'entier recherche existe dans la ligne$"

msg55 db "Element existe dans le fichier $"

msg6 db "\*Veuillez saisir un nombre,SVP : $", 0Dh,0Ah

msg7 db 0Dh,0Ah, "Le nombre en binaire est: $"

y db ?

.code

;===============================================

;==================================

;trois macro:couleur et affichage de l'entéte

;==================================

;===============================================

**attente macro**

mov ah,7

int 21h

**attente** endm

**aff\_msg macro chn** ;fonction pour afficher les messages

push dx

push ax

lea dx,chn

mov ah,9

int 21h

pop ax

pop dx

**aff\_msg** endm

**estitique macro a,c,b**  ;fonction permet d'afficher les titres et les entétes

mov bx,c

xor al,al

mov ah,9

mov cx,b

int 10h

lea dx,a

int 21h

**estitique** endm

;===================================

;ROUTINE:saisie le nom de fichier

;===================================

**saisie\_nom\_fichier proc**

mov al,nom\_fichier[1]

add al,2

xor ah,ah

mov bx,ax

mov nom\_fichier[bx],00h

mov al,2

mov ah,3dh ;3dh sert a ouvrir un fichier

mov dx,(offset nom\_fichier)+2

int 21h

mov handle2,ax ;handle2 :on l utilise pour accéder au fichier(méthodes pour accéder aux fichiers)

jc erreur1

jmp suivant

erreur1: ;cas d erreur d ouverture de fichier

aff\_msg msg2

JMP EXIT

suivant:

ret

**saisie\_nom\_fichier** endp

;=====================================

;MACRO:lecture de fichier

;=====================================

**ecrirfichier macro handle2,entier**

push cx

mov ah,40h ; 40f :Ecriture dans un fichier

mov bx,handle2

mov cx,6

mov dx,offset entier

int 21h

pop cx

**ecrirfichier** endm

;=====================================

;MACRO:lecture de fichier

;=====================================

**lecturefichier macro handle2,entier**

push cx

mov ah,3fh ; 40f :Ecriture dans un fichier

mov bx,handle2

mov cx,6

mov dx,offset entier

int 21h

pop cx

**lecturefichier** endm

;=================================================

;MACRO:remplissage de fichier par de code binaire

;=================================================

**remplir macro handle2,entier**

mov cx,0

lectur:

aff\_msg alaligne

mov ah,0ah ; 0ah(aussi 0ch) fait appel à une saisie au clavier.

mov dx,offset entier

int 21h

cmp entier[1],4 ; '4:Taille effective de le chaine /ENtier sera ecrit sur 4 bits..

jne message\_erreur

cmp entier[2],'0'

je passe

cmp entier[2],'1'

je condit

condit:

cmp entier[3],'0'

jne erreur

cmp entier[4],'0'

jne erreur

cmp entier[5],'0'

jl erreur

cmp entier[5],'1'

jg erreur

passe:

cmp entier[3],'0'

jl erreur

cmp entier[3],'1'

jg erreur

cmp entier[4],'0'

jl erreur

cmp entier[4],'1'

jg erreur

cmp entier[5],'0'

jl erreur

cmp entier[5],'1'

jg erreur

ecrir:

ecrirfichier handle2,entier

jmp suivant1

erreur: ;err si le chiffre de chaque entier est diff de 1 ou 0

aff\_msg message\_entier

message\_erreur: ;err si la taille de l entier est diff de 4 bit

aff\_msg mesg\_err

suivant1:

**remplir** endm

;==========================================

;MACRO: saisie de l'entier codé en decimal

;==========================================

**saisir macro somme,msg6**

aff\_msg msg6

faire:

mov ah,1

int 21h ;entier decimal entre O et 9

mov [somme],al ; car le fichier comporte que entre 0 et 9

cmp al,'9'

jg faire

cmp al,'0'

jl faire

aff\_msg alaligne

**saisir** endm

;=================================================

;MACRO: La trasformation en binaire

;=================================================

**transforme macro entier,somme,entier2,msg7**

mov cl,somme

;macro de tranforamtion decimale-binaire

mov si,offset entier2

mov bx,4

mov ax,cx

;sub ax,48

mov cx,4

calcul:

mov bl,2 ; 2 est la base de calcul

div bl

push ax

add ah,48

mov byte ptr[si],ah ;byte ptr:la taille(1 octet=1 bit) de la variable dans

inc si ; laquelle on a stocké le nombre

dec cx

pop ax

cmp cx,0

jne calcul

mov byte ptr[si],'$'

mov bx,3

mov cx,0

mov si,offset entier

saisi:

mov al,entier2[bx]

mov byte ptr[si],al

inc si

dec bx

inc cx

cmp cx,4

jne saisi

mov byte ptr[si],'$'

aff\_msg msg7

mov ah,9

mov dx,offset entier

int 21h

**transforme** endm

;============================================================

;MACRO: comparaison de l entier saisie avec celle de fichier

;============================================================

**transforme macro entier, entier2,msg5**

mov si,offset entier ;on compare notre entier à celle dan le fichier

mov bx ,0

compare:

mov al,byte ptr[si] ;byte ptr:la taille(1 octet=1 bit) de la variable dans

;mov ah,chaine2[bx] ; laquelle on a stocké le nombre

cmp al,entier2[bx]

jne suivant2

inc si

inc bx

cmp bx,4

je suivant11

jne compare

suivant11:

aff\_msg msg5

mov al,cl

add al,48

mov dl,al

mov ah,6

int 21h

suivant2:

**transforme** endm

;======================================================

;MACRO: recherche dans le fichier

;======================================================

**recherche macro handle2,entier2,entier,y**

mov ah,42h

mov al,0

mov bx,handle2

mov cx,0

mov dx,0

int 21h

mov cx,1

rech:

push cx

mov ah,3fh

mov bx,handle2

mov cx,4

mov dx,offset entier

int 21h

pop cx

comparaison entier,entier2 ,msg5

push cx

mov ah,42h

mov al,1

mov bx,handle2

mov cx,0

mov dx,2

int 21h

pop cx

inc cx

mov al,cl

cmp al,y

jne rech

aff\_msg msg55

**recherche** endm

;===============================================

;===============================================

;====PROGRAMME PRINCIPALE=======================

;===============================================

;===============================================

**main proc**

mov ax,@data

mov ds,ax

estitique msg10,85h,77

aff\_msg alaligne

estitique msg11,6,70

aff\_msg alaligne

estitique msg12,4,70

aff\_msg alaligne

aff\_msg nomfichier

aff\_msg alaligne

mov ah,0ah

lea dx,nom\_fichier

int 21h

**call saisie\_nom\_fichier** ;appel à la routine saisie\_nom\_fichier

aff\_msg alaligne

aff\_msg msg15

aff\_msg alaligne

aff\_msg alaligne

aff\_msg msg1

mov si,0

pour:

cmp si,5

je saut

**remplir handle2,entier,@y**

inc si

jmp pour

saut:

aff\_msg alaligne

**lecturefichier handle2,entier**

aff\_msg alaligne

**saisir somme,msg6**

**transforme entier,somme,entier2,msg7**

aff\_msg alaligne

**recherche handle2,entier2,entier,y**

EXIT:

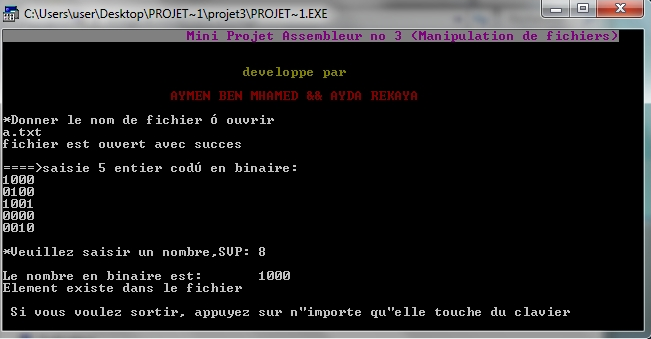
aff\_msg alaligne

estitique msg8,0ch,70

attente

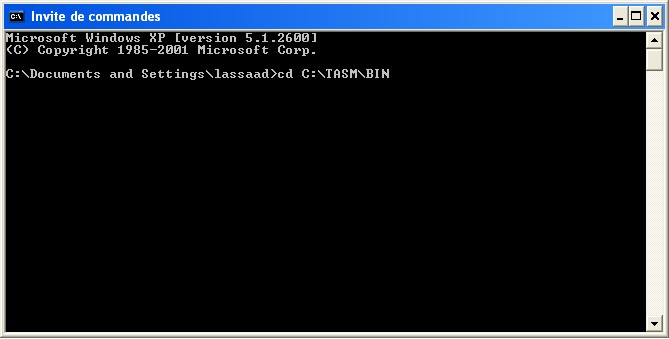
main endp

**end main**

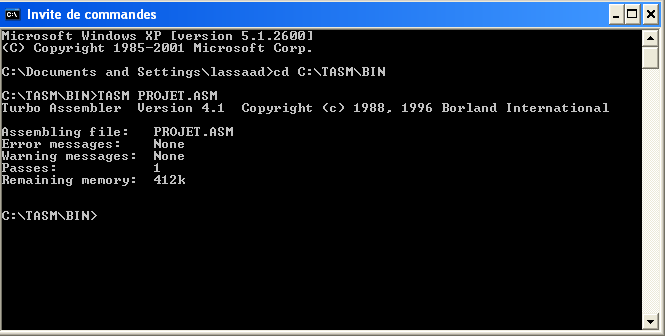


Étape de passage du code source en code exécutable

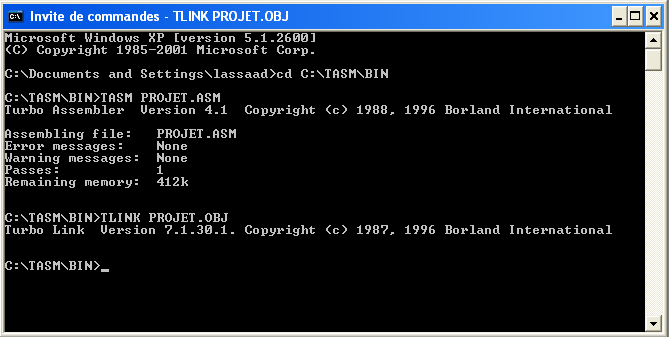
*Pour créer un programme assembleur il faut :*

* *Créer le programme au langage assembleur avec l’extension ″ .ASM ″.*
* *Ouvrir l’invite de commande (MS-DOS) pour effectuer les taches de compilations et de gestion des liens.*
* *En compilant avec l’utilisation de l’instruction ″tasm projet.asm″, le logiciel de compilation vérifie les erreurs et il les localise dans le programme, si le compilateur ne trouve aucune erreur dans le programme alors il crée un fichier portant le même nom avec l’extension ″ .obj ″ qui porte un programme presque converti en code exécutable ( ne converti pas les instruction d’appel des fonctions externes au programme source principal).*
* *La gestion des liens se fait par la balise ″tlink projet.obj″ qui assure la liaison entre les liens des fonctions utilisées dans le programme source et les fonctions définies ailleurs d’où il crée un fichier d’extension ″.map″ pour assurer la liaison les liens puis il crée le fichier exécutable.*

Changement de répertoire vers ″ C:\TASM\BIN ″



Compilation par ″TASM projet.asm ″ et création de fichier ″ PROJET.OBJ ″



Gestion des liens par ″ TLINK projet.obj ″ et création de fichier ″ projet.map ″ et d’exécutable ″ projet.exe ″

***QUESTION :***

***COMMENT ORGANISER LE FICHIER TEXTE POUR ACCELERER LA RECHERCHE ?***

***SOLUTION :***

***UTILISER LA METHODE DE TRIE POUR ORGANISER LE CONTINU D’UN FICHIER SELON UN ORDRE DONNEE .DANS CE SENS ON ACCELERE LA RECHERCHE***

CONCLUSION

*Lorsque on termine notre travail sur un fichier on doit le fermé .  
Si on ne le ferme pas ,un problème peut être lancé .*

*Attention lorsque vous laissez le fichier ouvert longtemps afin d'y ajouter*

*Progressivement des données ! Si le système plante, vous perdrez les données qui se trouvent*

*Dans le tampon à ce moment. C'est pourquoi il est conseillé de forcer régulièrement l'écriture sur le disque en refermant le fichier*.