# Fichiers

## I3 - Algorithmique et programmation

Nicolas Delestre

# Plan



Architecture des Systèmes d'Information

# Introduction

## Remarques

- Lorsque l'on utilise un programme :
  - on saisit des informations,
  - on « calcule » des nouvelle informations
- À l'arrêt du programme ces informations sont perdues

- L'objectif des fichiers est de conserver ces informations d'une exécution à l'autre du programme
- Cela amène à **enregistrer** de l'information et à **relire** de l'information
- C'est le troisième tiers d'une architecture classique trois tiers

Architecture des Systèmes d'Information

# Qu'est ce qu'un fichier ?

## Définition

- Un fichier est un ensemble d'informations enregistré sur un support de masse (disque dur, clé USB, etc.)
- Un fichier possède des métadonnées (nom, date de création, date de dernière modification, propriétaire, etc.)
- Les fichiers dans un ordinateur sont gérés par un système de fichiers (l'un des composants des systèmes d'exploitation). Ils sont rangés dans des répertoires (ou dossier)
- Les fichiers possèdent un format qui indique comment sont organisées les informations. Souvent le format apparaît par une extension suffixe du nom du fichier (par exemple `.txt`, `.pdf`, `.docx`, `.odt`)

asi
Architecture des Systèmes d'Information

# Représentation de l'information 1 / 3

## Tout est octet

- Lorsque l'on déclare une variable, sa valeur est stockée en mémoire de l'ordinateur suivant un certain codage
- Il existe des codages pour chaque type de données
- Ces codages peuvent être ou ne pas être normalisés
- Ces codages sont difficilement compréhensibles par un humain
  - `1100 0010 1110 1101 0100 0000 0000 0000` représente le nombre flottant $-118,625$ suivant la norme IEEE 754 (Wikipédia)
- Un codage peut être à taille fixe ou à taille variable
- La représentation textuelle est un codage humainement compréhensible. Plusieurs types de codage sont disponibles : ASCII, ASCII étendu (non normalisé), ISO 8859 - [1-15], Unicode
  - Les 32 premiers caractères de l'ASCII sont non imprimables
  - Attention l'interprétation de certains caractères non imprimables changent en fonction du système d'exploitation

# Représentation de l'information 2 / 3

## Naturel sur 8 bits

- De $0_{10}$ (00000000) à $255_{10}$ (11111111)

## Entier en complément à 1 sur 8 bits

- Méthode : comme pour les naturels. On verse les chiffres pour obtenir les entiers négatifs
- De $-127_{10}$ (10000000) à $127_{10}$ (01111111)
- Deux codages pour 0, 00000000 et 11111111

## Entier en complément à 2 sur 8 bits

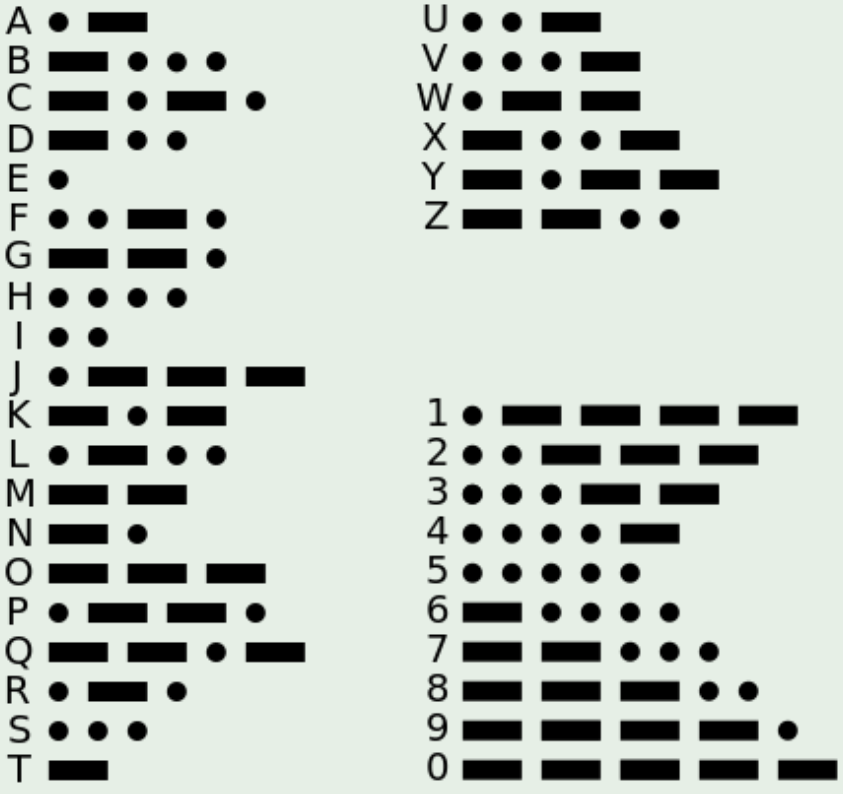
- Méthode pour le complément à 1 mais on ajoute en plus 1 pour les entiers négatifs
- De $-128_{10}$ (10000000) à $127_{10}$ (01111111)

# Représentation de l'information 3 / 3

## Morse

### Code morse international

1. Un tiret est égal à trois points.
2. L'espacement entre deux éléments d'une même lettre est égal à un point.
3. L'espacement entre deux lettres est égal à trois points.
4. L'espacement entre deux mots est égal à sept points.




Exercice : Que signifie ?

.- ... .. -.-. . ... - -... .. . -.

# Les fichiers en programmation 1 / 4

## L'assignation

- Un fichier est externe à un programme, il est identifié par son nom
- Un programme (dans la paradigme de la programmation structurée) manipule uniquement des variables
- Il faut associer une variable à un fichier. Nous verrons par la suite qu'il existe plusieurs types de variable permettant de manipuler différemment les fichiers. La variable contient différentes informations (nom du fichier, taille du fichier, position du curseur, etc.)
- Les actions sur la variable ont des actions sur le fichier (tant sur le contenu que sur la position courante du curseur)

## Attention

- De manière abusive on utilise le terme « fichier » aussi bien pour le fichier à proprement parlé que pour la variable qui permet de l'utiliser

# Les fichiers en programmation 2 / 4

## La notion de curseur

- Historiquement, abstraction de la position de la tête de lecture/écriture des unités de masse
- Indique la position de la prochaine lecture ou écriture
- L'écriture ou la lecture « fait avancer le curseur »

## Fichier à accès séquentiel

- On ne peut avoir accès à la $i^{eme}$ information, qu'après avoir lu les $i-1$ informations précédentes

## Fichier à accès aléatoire

- On peut accéder directement à la $i^{eme}$ information en déplaçant directement le curseur

Architecture des Systèmes d'Information

# Les fichiers en programmation 3 / 4

## L'ouverture, la fermeture

- Avant de pouvoir utiliser un fichier, il doit être ouvert. Il existe généralement quatre *mode* d'ouverture
  - lecture (position du curseur en début de fichier)
  - écriture (position du curseur en début de fichier)
  - lecture / écriture (position du curseur en début de fichier)
  - ajout (position du curseur en fin de fichier)
- Une fois que l'on a fini de l'utiliser, un fichier doit être fermé

# Les fichiers en programmation 4 / 4

## Actions

- savoir si le curseur est en fin de fichier
- lire (fichier ouvert en lecture, lecture/écriture) : permet de lire une information à la position du curseur (précondition : le curseur ne doit pas être en fin de fichier)
- écrire (fichier ouvert en écriture, lecture/écriture et ajout) : permet d'écrire une information à la position du curseur. Si le curseur est en fin de fichier alors le fichier est agrandi, sinon il y a écrasement des informations précédentes
- déplacer le curseur

# Les fichiers en Pascal 1 / 2

## Type des variables fichiers

- Il existe trois types pour les variables permettant de manipuler les fichiers :
  - `File` : fichier non typé
  - `File of` : fichier typé
  - `Text` : fichier texte

## Fonctions, procédures indépendantes du type

- système de fichiers (unité `sysutils`) : `fileExist`, `fileGetAttr`, `fileGetDate`, etc.
- assignation : `assign`
- ouverture : `reset`, `rewrite`
- fermeture : `close`
- fin de fichier : `eof`

# Les fichiers en Pascal 2 / 2

## Fonctions, procédures pour fichier binaire non typé

- lecture, écriture : `blockRead`, `blockWrite`
- gestion du curseur : `seek`, `filePos`, `fileSize`

## Fonctions, procédures pour fichier typé

- lecture, écriture : `read`, `write`
- gestion du curseur : `seek`, `filePos`, `fileSize`

## Fonctions, procédures pour fichier texte

- ouverture : `append`
- lecture, écriture : `read`, `readln`, `write`, `writeln`
- gestion du curseur : `seekEOF`, `seekEOLn`
- gestion du buffer : `flush`

# Écrire 42 1 / 3

## Premier exemple

- Écrire le contenu d'une variable entière (type Entier) qui vaut 42 dans un fichier :
  - Texte
  - Binaire (typé et non typé)

## Programme principal

```
var
   a : Entier;

begin
   a:=42;
   enregistrementTexte(a,'quaranteDeux.txt');
   enregistrementBinaireNonType(a,'quaranteDeux.bin1');
   enregistrementBinaireType(a,'quaranteDeux.bin2')
end.
```

# Écrire 42 2 / 3

```
procedure enregistrementTexte(a : Entier; nomFichier : String);
var
   fichierTexte : Text;
begin
   assign(fichierTexte,nomFichier);
   rewrite(fichierTexte);
   write(fichierTexte,a);
   close(fichierTexte)
end;

procedure enregistrementBinaireNonType(a : Entier; nomFichier : String);
var
   fichierBinaire : File;
begin
   assign(fichierBinaire,nomFichier);
   rewrite(fichierBinaire,1);
   blockWrite(fichierBinaire,a,sizeof(a));
   close(fichierBinaire)
end;

procedure enregistrementBinaireType(a : Entier; nomFichier : String);
var
   fichierBinaire : File of Entier;
begin
   assign(fichierBinaire,nomFichier);
   rewrite(fichierBinaire);
   write(fichierBinaire,a);
   close(fichierBinaire)
end;
```

# Écrire 42 3 / 3

## Exécution dans le cas où Type Entier = Byte

```
$ ls -al quaranteDeux.[bt]*
-rw-rw-r-- 1 ... 1 ... quaranteDeux.bin1
-rw-rw-r-- 1 ... 1 ... quaranteDeux.bin2
-rw-rw-r-- 1 ... 2 ... quaranteDeux.txt
$ cat quaranteDeux.txt
42$ hexdump quaranteDeux.bin1
0000000 002a
0000001
```

## Exécution dans le cas où Type Entier = Longword

```
$ ls -al quaranteDeux.[bt]*
-rw-rw-r-- 1 ... 4 ... quaranteDeux.bin1
-rw-rw-r-- 1 ... 4 ... quaranteDeux.bin2
-rw-rw-r-- 1 ... 2 ... quaranteDeux.txt
$ cat quaranteDeux.txt
42$ hexdump quaranteDeux.bin1
0000000 002a 0000
0000004
```

# Afficher le contenu d'un fichier texte 1 / 2

## Deuxième exemple

- Écrire un programme qui affiche le contenu d'un fichier (en considérant que c'est un fichier texte)

## Le programme

```
program afficherFichierTexte;

uses sysutils;

procedure afficherFichier(nomFichier : String);
var
   fichier : Text;
   uneLigne : String;
begin
   assign(fichier,nomFichier);
   reset(fichier);
   while not eof(fichier) do
   begin
      readln(fichier,uneLigne);
      writeln(uneLigne)
   end
end;
```

# Afficher le contenu d'un fichier texte 2 / 2

## Le programme (suite)

```
begin
   if ParamCount<>1 then
      writeln('Erreur : Un seul parametre = nom du fichier a afficher')
   else
      if not fileExists(ParamStr(1)) then
         writeln('Erreur : fichier inexistant')
      else
         afficherFichier(ParamStr(1))
end.
```

# Contacts 1 / 9

## Troisième exemple

- Écrire un programme de carnets de contacts téléphoniques permettant :
  - d'ajouter un contact
  - d'afficher les contacts

## Exemple d'utilisation

```
$ ./contacts test.fic "Torvald" "Linus" "(33)649785135"
$ ./contacts test.fic "Gates" "Bill" "(01)479520541657"
$ ./contacts test.fic
Nom : Torvald
Prenom : Linus
Telephone : (33)649785135
Nom : Gates
Prenom : Bill
Telephone : (01)479520541657
```

# Contacts 2 / 9

## Unité personne

```
unit Personne;

interface

type TPersonne = record
   nom : String;
   prenom : String;
   telephone : String;
end;

function personne(nom,prenom,telephone : String) : TPersonne;
function obtenirNom(p : TPersonne) : String;
function obtenirPrenom(p : TPersonne) : String;
function obtenirNumeroTelephone(p : TPersonne) : String;
```

# Contacts 3 / 9

## Unité personne (suite)

```
implementation

function personne(nom,prenom,telephone : String) : TPersonne;
begin
   personne.nom:=nom;
   personne.prenom:=prenom;
   personne.telephone:=telephone
end;

function obtenirNom(p : TPersonne) : String;
begin
   obtenirNom:=p.nom
end;

function obtenirPrenom(p : TPersonne) : String;
begin
   obtenirPrenom:=p.prenom
end;

function obtenirNumeroTelephone(p : TPersonne) : String;
begin
   obtenirNumeroTelephone:=p.telephone
end;

end.
```

# Contacts 4 / 9

## Unité contacts

```
unit contacts;

interface

uses personne;

type
   TTraiterPersonne = procedure(p : TPersonne);

procedure ajouterContact(nomFichier : String; p : TPersonne);
procedure parcourirContacts(nomFichier : String; traiter : TTraiterPersonne);
```

# Contacts 5 / 9

## Unité contacts (suite))

```
implementation

uses sysutils;

procedure ajouterContact(nomFichier : String; p : TPersonne);
var
   fichier : File of TPersonne;
begin
   assign(fichier,nomFichier);
   if fileExists(nomFichier) then
   begin
      reset(fichier);
      seek(fichier,fileSize(fichier))
   end
   else
      rewrite(fichier);
   write(fichier,p);
   close(fichier)
end;
```

# Contacts 6 / 9

## Unité contacts (fin)

```
end;

procedure parcourirContacts(nomFichier : String; traiter : TTraiterPersonne);
var
   fichier : File of TPersonne;
   p : TPersonne;
begin
   assign(fichier,nomFichier);
   if fileExists(nomFichier) then
   begin
      reset(fichier);
      while not eof(fichier) do
      begin
         read(fichier,p);
         traiter(p)
      end;
      close(fichier)
   end
end;
```

# Contacts 7 / 9

## Programme contactsTxt

```
program contactsTxt;

uses personne,contacts;

procedure afficherPersonne(p : TPersonne);
begin
   writeln('Nom : ',obtenirNom(p));
   writeln('Prenom : ',obtenirPrenom(p));
   writeln('Telephone : ',obtenirNumeroTelephone(p));
end;

procedure afficherContacts(nomFichier : String);
begin
   parcourirContacts(nomFichier,@afficherPersonne)
end;

procedure afficherAide();
begin
   writeln('contacts nom_fichier : permet d''afficher l''ensemble des contacts');
   writeln('contacts nom_fichier nom prenom telephone : permet d''ajouter un contact')
end;
```

# Contacts 8 / 9

## Programme contactsTxt (suite et fin)

```
begin
   case ParamCount of
      1 : afficherContacts(ParamStr(1));
      4 : ajouterContact(ParamStr(1),
                         Personne.personne(ParamStr(2),
                                           ParamStr(3),
                                           ParamStr(4))
                        );
      otherwise afficherAide()
   end
end.
```

# Contacts 9 / 9

## Exercices

- Améliorer ce programme de façon à pouvoir :
  - n'afficher que les contacts d'un certain nom
  - supprimer un contact
  - afficher les contacts en ordre alphabétique