•Algorithmique & Langage C

•IUT GEII S1

•

•Les fichiers

•

•Notes de cours

• (septième partie)

7

cours_algo_lgc7.09.odp


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Licence


•Paternité - Pas d'Utilisation Commerciale -
• Partage des Conditions Initiales à l'Identique 2.0 France

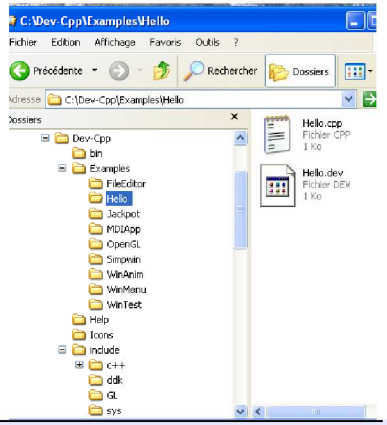
•Vous êtes libres :
• * de reproduire, distribuer et communiquer cette création au public
• * de modifier cette création, selon les conditions suivantes :
•
•**Paternité**. Vous devez citer le nom de l'auteur original.
•
•**Pas d'Utilisation Commerciale**.
•Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.
•
•**Partage des Conditions Initiales à l'Identique**.
•Si vous modifiez, transformez ou adaptez cette création,
•vous n'avez le droit de distribuer la création qui en résulte
•que sous un contrat identiqueà celui-ci.
• * A chaque réutilisation ou distribution, vous devez faire apparaître clairement aux autres les
• conditions contractuelles de mise à disposition de cette création.
•* Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droit
•Ce qui précède n'affecte en rien vos droits en tant qu'utilisateur (exceptions au droit d'auteur :
•copies réservées à l'usage privé du copiste, courtes citations, parodie...)
•voir le contrat complet sous : http://fr.creativecommons.org/contrats.htm


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Fichier, chemin de fichier

Un **fichier** est vu comme une suite d'octets dans la taille n'est pas limitée a priori.

Un **répertoire** (dossier, folder, directory) est un fichier qui contient des références vers d'autres fichiers ou des répertoires.

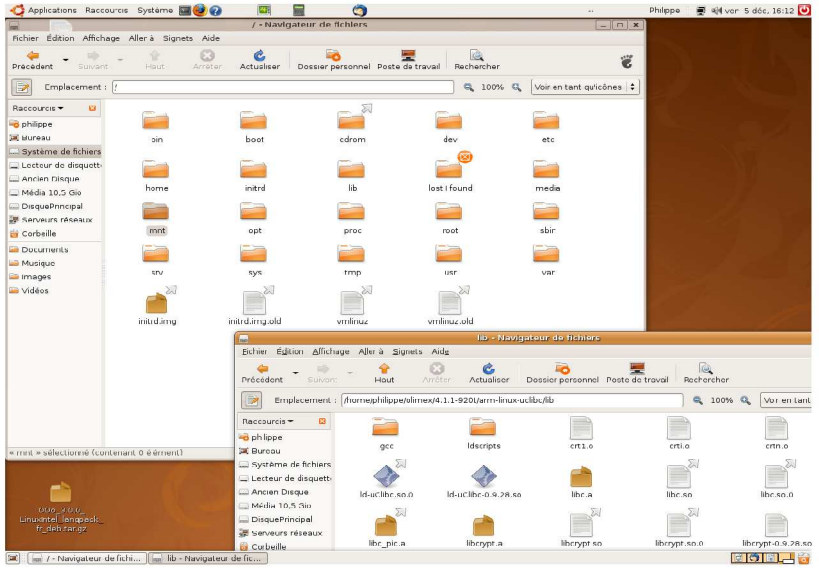


Les fichiers sont organisés de manière hiérarchique. Le point d'entrée de cette hiérarchie est la **racine** du système de fichier.

Ici la racine est **c:**
Sous linux elle est notée **/**

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Fichiers, chemins de fichiers



© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Chemin de fichier

Un **chemin de fichier** est une liste de répertoires séparées par le symbole **/** et terminée par un nom de fichier.

Un chemin permet de désigner un fichier dans l'arborescence.

Exemples : `c:/Dev-Cpp/Examples/Hello/Hello.cpp`
`include/stdc/stdio.h`

Si le chemin se termine par un nom de répertoire, il s'agit d'un *chemin de répertoire*.

`c:/Dev-Cpp/Examples/Hello`
`include/stdc`
`include/stdc/`


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Ne pas utiliser la notation Microsoft©

Attention seul Microsoft© note les chemins avec **** à la place **/** !
Partout ailleurs (url web, linux, unix, langage C, etc..)
c'est **/** qui est utilisée.

`c:/Dev-Cpp/include/stdio.h` normal

`c:\Dev-Cpp\include\stdio.h` Microsoft©

Il ne faut pas utiliser la notation Microsoft© en langage C.

En effet l'utilisation de **** dans une chaîne de caractère conduit à une interprétation de certains caractères :

"c:\toto\nanar" \t tabulation , \n saut de ligne ....


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Chemin absolu, chemins relatifs

Lorsque le premier répertoire du chemin est le *répertoire racine* le chemin est dit **absolu**. Un chemin absolu désigne un fichier de manière *unique* dans l'arborescence.

```
/usr/phil/tpc/tp1.c
c:/etudiants/rapports/rapport1.odt
```

Lorsque on ne part pas de la racine, le chemin est dit **relatif**. Un chemin relatif n'a de sens que par rapport à un répertoire donné. Il existe donc plusieurs chemins relatifs pour désigner un même fichier ou un même répertoire.

Dans un chemin relatif la notation :

- `.` désigne le répertoire courant
- `..` désigne le répertoire contenant le répertoire courant

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Chemins relatifs (cf figure page 35)

Le fichier `c:/Dev-Cpp/include/c++/3.4.2/bacward/algo.h` est désigné par les chemins :

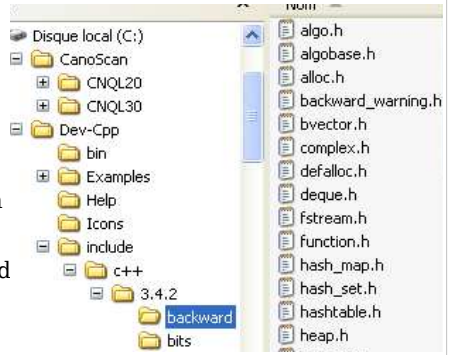
`c++/3.4.2/backward/algo.h` depuis include

`backward/algo.h` depuis 3.4.2

`../bacward/algo.h` depuis bits

`../include/c++/algo.h` depuis bin

`./algo.h ou algo.h` depuis bacward



`../../Dev-Cpp/include/c++/3.4.2/bacward/algo.h` depuisCNQL20

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Extension

L'extension d'un fichier est la suite de caractères qui suit le dernier point dans le nom d'un fichier.

`rapport1.odt` porte l'extension **`odt`**

`toto.txt.exe` porte l'extension **`exe`**

Le système d'exploitation peut associer l'exécution de certains programme à certaines extensions :
txt -> éditeur de texte , jpg -> afficheur d'image, etc...

Windows© réserve certaines extensions pour certains types de fichiers : exe -> programme , bat -> script, ...

Pour linux l'extension d'un fichier peut être quelconque. Les droits d'exécutions ne dépendent pas du nom du fichier.

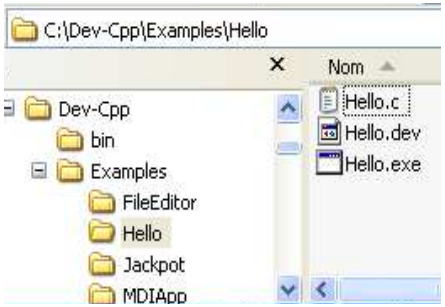
© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Notion de répertoire courant

Le répertoire courant d'un programme est le répertoire de référence des chemins relatifs pour ce programme.

Par défaut, c'est le répertoire dans lequel se trouve le fichier exécutable du programme.

`c:/Dev-Cpp/Examples/Hello/`
est le répertoire courant
du programme **`Hello.exe`**



© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Accès aux fichiers en C

Pour accéder à un fichier en C, il faut d'abord l'**ouvrir**. L'opération d'ouverture fait le lien entre un fichier désigné par son chemin dans l'arborescence et une variable du programme appelée **descripteur de fichier**.

Le descripteur de fichier sera ensuite passé en paramètre de chaque fonction d'écriture ou de lecture dans le fichier.

Type d'un descripteur : `FILE *`

(définit dans `stdio.h`)

Après l'accès, le fichier sera **fermé** par le programme.


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Écrire dans un fichier

Déclaration d'un descripteur : `FILE * fichier ;`

Ouverture : `fichier = fopen( "toto.txt" , "w" ) ;`

Cette opération :

1/ Crée un fichier vide dont le chemin est `toto.txt`.
On peut utiliser, soit le chemin absolu,
soit le chemin relatif par rapport au répertoire courant.

2/ Associe le descripteur `fichier` au fichier `toto.txt`

C'est la seule ligne où va apparaître le chemin du fichier.
Ensuite c'est le descripteur qui sera utilisé pour accéder au fichier.


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Écrire dans un fichier

L'opération d'ouverture d'un fichier peut échouer :
média protégé en écriture, disque plein, pas de droit en écriture dans le répertoire, lecteur absent,...

Dans ce cas **fopen** retourne la valeur **NULL** (descripteur invalide). Il ne faut alors pas tenter d'écrire sinon le programme plante.

```
if  (fichier != NULL ) {
                // écriture possible
             }
     else
              {
                 //  échec d'ouverture
                 //  écriture impossible
              }
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Écrire dans un fichier

Après avoir ouvert le fichier et n'avoir pas obtenu **NULL**, on peut utiliser la fonction **fprintf** pour écrire dedans.

fprintf est identique à **printf** sauf que l'on passe en premier paramètre le descripteur du fichier dans lequel on souhaite écrire :

```
int x = 4 ;
char c = 'A' ;
fprintf ( fichier , "bonjour !\n" ) ;
fprintf ( fichier , " x=%d c=%c", x, c ) ;
```

On peut aussi utiliser **fputc** pour écrire un seul caractère :

```
fputc ( c  ,  fichier ) ;
fputc ( 'Z' , fichier  ) ;
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Écrire dans un fichier

```
FILE * fichier ;

fichier = fopen( "toto.txt" , "w" ) ;

if ( fichier != NULL ) {


        fprintf( fichier , "Bonjour !\n" ) ;
        fclose( fichier );
}
else {
       printf("Erreur d'ouverture du fichier!\n");
     }
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Écrire dans un fichier

L'écriture dans un fichier est la plupart du temps *bufferisée* :

Les octets à écrire sont mis en attente dans un buffer
et le buffer est inscrit sur le disque lorsque :

- il est plein (1 à 2 ko) (mode fully buffered)
- il est plein on rencontre un caractère \n (line buffered)

Si le programme "plante" avant l'écriture effective, les octets présents dans le buffer sont perdus.

Pour forcer l'écriture sur le disque : **`fflush ( fichier ) ;`**

La fonction fclose provoque le vidage du buffer avant de fermer le fichier : **`fclose ( fichier ) ;`**


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Modes d'ouverture

Le deuxième paramètre de fopen est une chaîne de caractère appelée **mode d'ouverture** :

"w" (write) : crée un fichier vide pour y écrire.
Si le fichier existe il est détruit. Attention !

"a" (append) : identique à **"w"** si le fichier n'existe pas.
Si le fichier existe, prépare l'écriture à la fin.

"r" (read) : ouvre le fichier en lecture.
Prépare la lecture du premier caractère du fichier.

Pour windows© uniquement, il peut être nécessaire de rajouter le suffixe b pour éviter une interprétation de certains caractères : **"wb"**, **"ab"** , **"rb"**


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lecture d'un caractère

Après avoir ouvert un fichier en lecture, on peut y lire le premier caractère. Un "curseur" imaginaire avance alors d'une position; la prochaine lecture permettra d'accéder au prochain caractère. Et ainsi de suite...

```
FILE * fichier ;
int c ;   // déclaration de c en int !!!

fichier = fopen( "toto.txt" , "r" ) ;

if ( fichier != NULL ) {
                        c = fgetc ( fichier ) ;
                        printf("1er  car : %c",c) ;
                        c = fgetc ( fichier ) ;
                        printf("2eme car : %c",c) ;
                        ....;   }
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Détection de la fin d'un fichier

Lorsqu'on tente de lire après le dernier caractère, on obtient une valeur notée **`EOF`** : **`End Of File`**

Comme les octets d'un fichier peuvent prendre toutes les valeurs possibles (de 0x00 à 0xFF), la valeur **`EOF`** ne peut pas être un octet.
C'est souvent la valeur entière -1 (0xFFFFFFFF sur 32bits).
Il faut donc toujours déclarer en `int` un caractère lu dans un fichier.

Si on le déclare en char, l'apparition de l'octet 0xFF dans un fichier sera considérée comme la fin !
Dans un fichier texte c'est peu fréquent car 0xFF correspond au caractère 'ÿ' en ISO-LATIN-1 mais un fichier quelconque peut contenir n'importe qu'elle valeur...

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lire jusqu'à la fin

Attention, la fin du fichier n'est pas détectée par la lecture du "dernier" caractère.
C'est une tentative de lecture au delà qui provoque la détection de la fin du fichier.

Il faut donc répéter tant qu'on n'a pas EOF :

1/ lire un caractère
2/ tester si on a obtenu EOF
3/ Si ce n'est EOF exploiter le caractère


```
c = fgetc( fichier );
```

Plus de caractères !

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lecture caractère par caractère

```
FILE * fichier ;

int c = !EOF ;   // déclaration de c en int !!!

fichier = fopen( "toto.txt" , "r" ) ;

if ( fichier != NULL ) {

while( c != EOF ) {

          c = fgetc(fichier);
          if ( c!=EOF ) {
             // exploiter c
            }
      }
     fclose ( fichier ) ;       }
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lecture caractère par caractère

```
FILE * fichier ;

int c  ;   // déclaration de c en int !!!

fichier = fopen( "toto.txt" , "r" ) ;

if ( fichier != NULL ) {

while( c = fgetc(fichier) , c != EOF ) {


            // exploiter c

 }
fclose ( fichier ) ;       }
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lire des valeurs dans un fichier

Le programme de lecture va dépendre de l'organisation du fichier à lire.

Considérons le fichier **`mesures.txt`** contenant sur chaque ligne un relevé de tension et de courant aux bornes d'une résistance :

```
3.200  0.234
3.124  0.229
3.356  0.245
.....  .....
```

On va lire ces valeurs et les stocker dans deux variables réelles comme on le ferait si elles étaient saisies au clavier.


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lire des valeurs dans un fichier

```
FILE * fichier ;
float u , i ;

fichier = fopen( "mesures.txt" , "r" ) ;

if ( fichier != NULL ) {

        fscanf( fichier , "%f", &u ) ;
        fscanf( fcihier , "%f", &i ) ;
        printf("u=%f i=%f r=%f\n", u , i , u/i );
        fclose ( fichier ) ;
}
else {
       printf("Erreur d'ouverture du fichier!\n");
     }
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

La fonction `fgets()`

Nous avons déjà utilisé cette fonction sur l'entrée standard.

Ici en passant en dernier paramètre un descripteur de fichier, on peut lire une chaîne à partir de la position courante dans un fichier :

```
char ch[160] ;

fgets ( ch , 160 , fichier ) ;
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lire des valeurs dans un fichier

Comme le fichier est organisé par ligne, on peut aussi lire une ligne et l'analyser ensuite :

```
FILE * fichier ;
float u , i ;
char ch[160];

fichier = fopen( "mesures.txt" , "r" ) ;

if ( fichier != NULL ) {

        fgets(  ch , 160 , fichier ) ;
        sscanf( ch , "%f %f", &u , &i ) ;
        printf("u=%f i=%f r=%f\n", u , i , u/i );
        fclose ( fichier ) ;
}
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lire jusqu'à la fin

Si on connait le nombre de valeurs, il suffit de répéter la lecture autant de fois. Mais en général le nombre de valeurs n'est pas connu a priori. Il faut donc lire les valeurs "tant qu'il y en a" : jusqu'à la fin du fichier.

La fonction **feof** retourne une valeur non nulle lorsque la fin d'un fichier est atteinte.

```
if ( feof ( fichier )  != 0 ) {
                    // fin atteinte   }
                else {
                      // il reste des valeurs
                     }
```

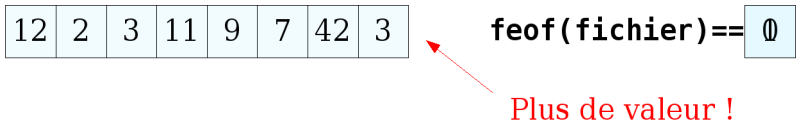

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lire jusqu'à la fin

Attention, la fin du fichier n'est pas détectée par la lecture de la "dernière" valeur.
C'est une tentative de lecture au delà qui provoque la détection de la fin du fichier.

Il faut donc répéter :

1/ lire
2/ tester la fin
3/ exploiter la lecture si ce n'est pas la fin




© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lecture ligne par ligne jusqu'à la fin

```
FILE * fichier ;
float u , i ;
int fin = 0 ;
char ch[160];
fichier = fopen( "mesures.txt" , "r" ) ;
if ( fichier != NULL ) {

while( fin==0 ) {

          fgets( ch , 160 , fichier ) ;
          fin = foef ( fichier ) ;
          if ( fin == 0 ) {
             // exploiter ch
            }
      }

    fclose ( fichier ) ;       }
```

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lecture ligne par ligne jusqu'à la fin

```
FILE * fichier ;
float u , i ;
char ch[160];

fichier = fopen( "mesures.txt" , "r" ) ;

if ( fichier != NULL ) {

while( fgets( ch,160,fichier ), foef(fichier)==0 ) {

          sscanf( ch , "%f %f", &u ) ;
          printf("u=%f i=%f r=%f\n", u , i , u/i );
       }

     fclose ( fichier ) ;
}
```

© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Déplacement dans un fichier

On peut atteindre directement une position donnée dans un fichier ouvert en lecture :

Déplacement de **noctets** par rapport au premier caractère :

```
fseek ( fichier , noctets , SEEK_SET ) ;
```

Déplacement de **noctets** par rapport à la position courante :

```
fseek ( fichier , noctets , SEEK_CUR ) ;
```

Déplacement de **noctets** par rapport au dernier caractère :

```
fseek ( fichier , noctets , SEEK_END ) ;
```


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Lecture/Ecriture de données brutes (binaires)

```
size_t fread (void * adr, size_t taille, size_t nblocs,
FILE * f);
```

Lit à la position courante sur le fichier `f`, `nblocs` de taille octets et les range à partir de l'adresse `adr`. Retourne le nombre de blocs réellement lus.

```
size_t fwrite(void * adr, size_t taille, size_t nblocs,
FILE * f);
```

Ecrit à la position f sur le fichier les `nblocs` de `taille` octets à partir de l'adresse `adr`.

Pour plus de précision voir : cours_de_c_v200.pdf chapitre 10.


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Exercices

- Ecrire un programme qui demande un nom sans extension pour un fichier, rajoute l'extension "txt", et crée ce fichier. Ensuite le programme écrira des nombres aléatoires compris entre -5 et 5 dans ce fichier.
- Ecrire un programme qui duplique un fichier. Le nom de la copie sera formé à partir du nom du premier auquel on rajoutera 2 avant l'extension. "toto.c" -> "toto2.c"
- Ecrire un programme qui compte le nombre d'occurrence du caractère 'e' dans un fichier.
  Modifier le programme pour compter le nombre d'occurrence de chaque caractère et stocker les résultats dans un fichier.


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Exercices

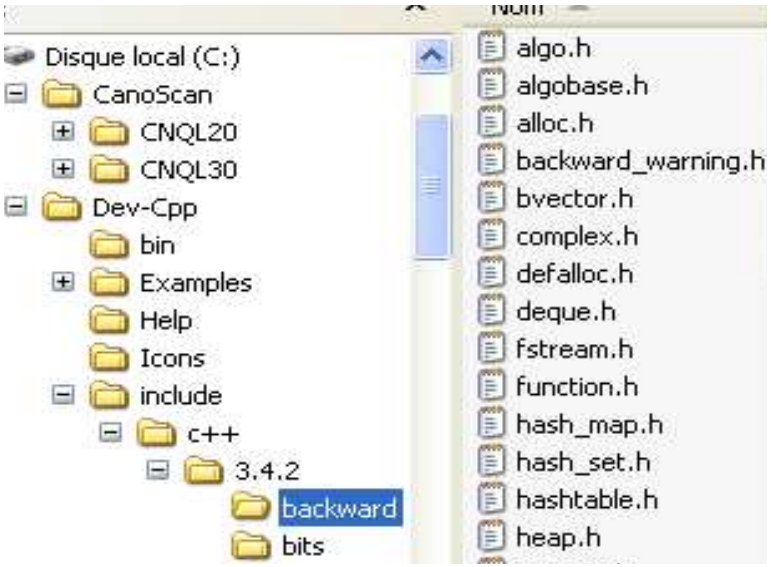
- Soit un fichier de mesures organisé de la manière suivante :

```
##Date : 23/12/06
##Nom : Albert Popov
##Tension   Courant
43.4V       23mA
46.5V       24.7mA
38.9V       20.7mA
....        .....
```

  Faire un programme qui affiche à l'écran la date de la mesure, le nom de l'opérateur et le nombre de mesures ainsi que la valeur maximale de la tension appliquée.
  Votre programme créera ensuite un fichier identique au précédent mais en rajoutant une colonne Puissance obtenue en faisant le produit des deux valeurs d'une même ligne.


© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Chemins relatifs



© Copyright 2006, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license