Université de Limoges 2004-2005
Faculté des Sciences

TP Algorithmique N° 1

L'objectif de ce premier TP est de découvrir l'environnement de travail qui sera utilisé lors des prochaines séances.

1 - Environnement de travail

Tous vos travaux s'effectueront à partir d'une console *xterm* qui vous servira à lancer les commandes. Pour lancer un terminal : alt+F2 "xterm" sous l'environnement KDE.

1.1 - Commandes Unix de manipulation de fichiers

Vous trouverez ci-dessous une liste de quelques commandes Unix. Les paramètres entourés de "<>" sont optionnels, tandis que les paramètres entourés de "[]" sont requis. Pour toutes ces commandes, vous pouvez accéder à une description de leurs très nombreuses commandes avec la commande *man*.

man

Description : Affiche les pages de manuels de la commande en paramètre.
Exemple : man man
Profil : man <options> [nom d'une commande unix]
Tips : Taper "/" suivi d'un mot pour rechercher l'occurence d'un mot dans la doc.
Taper "n" pour aller à l'occurence suivante.

mkdir

Description : Crée un nouveau répertoire.
Exemple : mkdir glop
Profil : mkdir [nom du répertoire à créer]

cd

Description : Entre dans le répertoire donné en paramètre.
Exemple : cd glop
Profil : cd <nom d'un répertoire>
Tips :

- la commande *pwd* permet d'afficher le répertoire actuel.
- la touche tabulation permet d'autocompléter les noms de répertoire ou de fichier.
  Ex: taper "cd g[TAB]" affichera "cd glop" automatiquement. (si plusieurs fichiers commencent par la lettre "g", ils sont listés.)
- si votre nom de répertoire contient un espace vide, le remplacer par "\ " (slash suivi d'un espace). Ex : cd glopi\ glopa
- *cd* sans paramètres vous ramène à la racine de votre compte. (tout comme *cd ~*)

ls

Description : Liste le contenu d'un répertoire (par défaut, le répertoire actif)
Exemple : ls
Profil : ls <options> <répertoire>
Tips : ls -l, ls -al, etc.. (*man ls* pour toutes les options.)

cp

Description : Source et destination peuvent être un fichier ou un répertoire. Dans l'exemple, le fichier *tp1.cpp* est copié dans le répertoire *glop*.
Exemple : cp tp1.cpp glop/
Profil : cp <options> [source] [destination]
Tips : *cp -R* [...] permet d'effectuer des copies récursivement. Ex : cp -R glop1/ glop2/ copiera tous les fichiers et répertoires contenus dans le répertoire glop1 dans le répertoire glop2/.

rm

Description : Supprime le fichier en paramètre.
Exemple : rm tp1.cpp
Profil : rm <options> [fichier ou répertoire]
Tips : *rm -R* [...] supprime un répertoire. Ex: rm -R glop/

mv

Description : Déplace un fichier ou un répertoire d'un emplacement à un autre.
Exemple : mv tp1.cpp glop/
Profil : mv <options> [source] [destination]

find

Description : Recherche un fichier (ou un répertoire) dont le nom contient "tp1", à partir de l'emplacement courant (dénoté par ".")
Exemple : find . -name "*tp1*"

tar

Description : Archivage des données.
Exemple : *tar zcvf sauvegarde_tp1.tgz glop/* sauvegardera le contenu du répertoire glop dans un fichier d'archive nommé *sauvegarde_tp1.tgz*
tar zxvf sauvegarde_tp1.tgz décompresse le fichier et restaure glop/

1.2 - Editeurs de texte

Différents logiciels sont à votre disposition pour créer les programmes qui vous seront demandés. Vous pourrez utiliser *kwrite,* e*macs* (ou mieux : *Vim*).

Emacs est un environnement d'édition complet qui dispose d'un très grand nombre de commandes, accessibles à l'aide des touches de contrôle. Les notations utilisées pour désigner ces touches sont les suivantes :

C-	désigne la touche contrôle (Ctrl)
M-	désigne la touche méta (Alt)
DEL	désigne la touche Backspace (retour arrière)
RET	désigne la touche Enter (entrée)
SPC	désigne la touche d'espace

ESC désigne la touche esc
TAB désigne la touche de tabulation

Par exemple, une combinaison "C-M-" (ou M-C-) pour Emacs correspond donc à la pression simultanée des touches Ctrl et Alt.
Exemple : l'ouverture d'un fichier s'effectue avec la commande C-X-f.

1.3 - Compilation et édition des liens : g++

La compilation du programme s'effectuera à l'extérieur de l'éditeur (sauf si vous utilisez Emacs ou Vim !). Nous utiliserons le compilateur g++ en ligne de commande pour découvrir les séquences de compilation et d'édition des liens.
Pour compiler un programme écrit dans le fichier *main.cpp*, taper : *g++ -c main.cpp*

► Le compilateur va analyser votre programme et afficher les erreurs éventuelles.

Une fois que le programme est compilé sans erreurs, vous pourrez vérifier qu'un fichier objet (d'extension .o) a été créé (ici, *main.o*).

L'édition des liens et la création de l'exécutable s'effectue par : *g++ -o main main.o*

► Ceci crée un fichier exécutable nommé *main* à partir du fichier objet compilé *main.o*.

Pour lancer l'exécutable, toujours en ligne de commande tapez : *./main*

► Votre programme se lance.

2 - Exercice

2.1 - Compilation avec g++

Recopiez le programme c++ donné page suivante dans un fichier nommé *gestion_records.cpp*.
Compilez ce fichier (*g++ gestion_records.cpp -o gestion*) et exécutez le (*./gestion*).

2.2 - Compilation séparée et édition des liens

Séparez *gestion_records.cpp* en différents fichiers (voir page suivante) correspondants aux actions de saisie, de tri et d'affichage. Ces fichiers seront compilés séparément.

Chaque action sera implémentée par une fonction. Leurs profils sont :

```
void saisie( athlete part[], int nbp );
void tri( athlete part[], int nbp );
void affichage( athlete part[], int nbp );
```

```cpp
#include <iostream>
using namespace std;

// Données caractéristiques d'un athlète
// -------------------------------------
struct athlete
{
   int   numero;
   float record;
};

// Nombre de participants
// ----------------------
const int nbParticipants = 10;

// Tableau d'athlètes
// ------------------
athlete participants[ nbParticipants ];

int main( void )
{
   athlete tmp;
   int     i, j, indiceMin, rang;
   float   min;

   // Saisie du tableau
   // -----------------
   for( i = 0; i < nbParticipants; i++ )
   {
      cout << "N° et record ?" << endl;
      cin >> participants[i].numero >> participants[i].record;
   }

   // Tri du tableau
   // --------------
   for( j = 0; j < nbParticipants - 1; j++ )
   {
      min = participants[ j ].record;
      indiceMin = j;

      for( i = j+1; i < nbParticipants; i++ )
         if( participants[ i ].record < min )
         {
            min = participants[ i ].record;
            indiceMin = i;
         }

      if( indiceMin != j )
      {
         tmp = participants[ j ];
         participants[ j ] = participants[ indiceMin ];
         participants[ indiceMin ] = tmp;
      }
   }

   // Affichage des résultats
   // -----------------------
   cout << "Rang Participant Record" << endl;
   cout << "1   " << participants[ 0 ].numero << "    " << participants[ 0 ].record << endl;

   rang = 1;
   for( i = 1; i < nbParticipants; i++ )
   {
      if( participants[ i ].record > participants[ i - 1 ].record )
         rang++; // On n'incrémente le rang que dans le cas ou il n'y a pas égalité

      cout << rang << "   " << participants[ i ].numero << "    " << participants[ i ].record << endl;
   }

   return 0;
}
```

gestion_types.h

gestion_saisie.cpp

gestion_tri.cpp

gestion_affichage.cpp