UNIVERSITE PAUL SABATIER

TOULOUSE III

TP Module 3 - Langage C

L2 IUP AISEM

2004/2005

J.M. ENJALBERT

Présentation de l'environnement de travail

Session de travail

Les séances de TP langage C se déroulent sur des machines fonctionnant sous LINUX dans l'environnement graphique KDE.

LINUX est un système d'exploitation multi-taches et multi-utilisateurs. Chaque utilisateur possède un compte identifié par un nom de login et protegé par un mot de passe. Il possède par ailleurs un répertoire personnel (espace personnel protégé sur le disque).

Les machines étant reliées en réseau (local!), l'accès a un compte utilisateur peut se faire indifféremment à partir de n'importe quelle machine.

La procédure permettant de travailler sur son compte consiste à:

- rentrer son nom de login à l'invite: `login:`
- rentrer son mot de pasee à l'invite: `password:` (attention, il n'y a pas d'écho a l'écran des touches actionnées)
- lancer l'environnement graphique en tapant: `startx`
- ouvrir une fenêtre terminal en cliquant sur l'icône adéquate.

(Toutes les entrées au clavier sont validées par la touche `enter`)

En fin de séance la procédure à suivre consiste à:

- sortir de l'environnement graphique
- se déconnecter en frappant sur la combinaison de touche CTRL-D
- arréter la machine à l'aide de la combinaison de touches CTRL-ALT-SUPPR (3 touches). (**Ne pas utiliser l'interrupteur marche/arrêt!**)

Un nom de login vous sera fourni à la première séance de TP. Il est de la forme: `l2aisem_??`. Le mot de passe par défaut est: `jsupe2004`. Vous pouvez le modifier en utilisant la commande `yppasswd`.

Outils pour la programmation en C

L'écriture, la compilation, la mise au point et l'exécution d'un programme C fait appel à différents outils.

Les instructions du programme doivent être rentrées comme du texte normal dans un fichier à l'aide d'un éditeur de texte. On parle de programme *source*. Le fichier contenant un programme source doit comporter l'extension .c (Ex: prog.c).

Ce fichier source doit ensuite être compilé pour créer un exécutable (fichier binaire) ou un fichier objet (d'extension .o). Ceci est fait en utilisant un *compilateur*.

L'exécution du programme consiste tout simplement à taper son nom dans une fenêtre terminal.

Ces outils (traitement de texte et compilateur) seront lancés a partir d'une fenêtre terminal sous la forme de commandes.

Edition du programme:

N'importe quel éditeur de texte pourrait faire l'affaire. En pratique, vous utiliserez l'éditeur de textes *emacs*.

Pour le lancer: taper `emacs &` dans la fenêtre terminal (ou trouver la commande dans le menu général).

Pour créer un nouveau fichier (programme): choisir `open` dans le menu *file* et taper son nom (Ex: prog1.c). Taper le programme dans la fenêtre d'édition. Pour sauver: *save* dans le menu *file* Des outils d'édition (couper, coller, etc...) sont disponibles (accessibles par le menu *edit* ou par des raccourcis claviers).

Attention: tout programme source devra être sauvé en lui donnant l'extension .c, par exemple *programme.c*

On peut aussi éditer un programme existant en utilisant l'item *open* du menu *file*.

Compilation:

Le compilateur C utilisé sera un compilateur GNU: *gcc*

Pour compiler, par exemple, le programme *programme.c* on tapera la ligne suivante:

```
gcc -Wall -g -o programme programme.c -lm
```

Si tout ce passe bien, l'exécutable *programme* sera créé qui pourra être lancé en tapant simplement `programme` dans la fenêtre terminal.

Impression des listings

L'impression des listings se fait sur une imprimante connectée au réseau. A partir de la ligne de commande il suffit de taper la commande `a2ps mon_fichier.c` pour sortir un listing formaté avec entête. La commande `a2ps` est en fait un *filtre* d'impression qui traduit un texte ascii en postscript en fonction de son type. Un fichier d'extension `.c` est ainsi reconnu comme un source C et traité en conséquence (commentaires en italiques, etc...).

Le contrôle de l'impression se fait en utilisant la commande `lpq` qui permet de visualiser les travaux d'impressions en attente. Chacun des travaux est associé a un numéro qui doit être utilisé pour l'enlever éventuellement de la file d'impression en invoquant la commande `lprm` et en lui passant ce numéro en argument.

Il est rappelé que l'imprimante est une ressource collective. Le respect des autres utilisateurs impose de vérifier que l'impression que l'on a lancée s'est bien déroulée et ne bloque pas l'imprimante pour une raison ou pour une autre. Le papier est par ailleurs une ressource rare qui doit donc être utilisée à bon escient.

En pratique deux imprimantes sont reliées au réseau. L'une d'entre elles est définie comme imprimante par défaut mais on peut accéder a l'autre en précisant son nom grace à l'option `-P`. Par exemple: `a2ps -Pimp1 prog.c` enverra le travail d'impression sur l'imprimante 1 (a droite en entrant) et `a2ps -Pimp2 prog.c` l'enverra sur l'imprimante 2. L'imprimante définie par défaut dépend de la machine sur laquelle vous travaillez.

Séance 1

(affectation, instruction if else, entrées/sorties clavier/écran, boucles for)

1.1 Premier programme

Ecrire un programme affichant a l'écran un message de bienvenue. Le compiler puis l'exécuter.

1.2 Conversion

Ecrire un programme qui lit un nombre au clavier (représentant une valeur en euros) et la convertit en francs. Rappel: 1 euro =6.55957 francs. On affichera le résultat avec deux chiffres après la virgule.

1.3 Equation du second degré

1.3.1

Le calcul des racines d'une équation du second degré du type $ax^2+bx+c=0$ se fait en calculant d'abord son discriminant $\Delta = b^2 - 4ac$.

Ecrire un programme permettant de calculer Δ et d'afficher sa valeur à l'écran pour des valeurs fixées de a, b et c.

1.3.2

Compléter le programme précédent de manière à pouvoir rentrer les valeurs de a, b et c au clavier. Compiler et tester.

1.3.3

Calculez et afficher les solutions de l'équation dans le cas ou $\Delta > 0$. ($x_1 = (-b + \sqrt{\Delta})/2a$ et $x_2 = (-b - \sqrt{\Delta})/2a$)

1.3.4

Traiter les cas $\Delta = 0$ (1 racine double: $x = -b/2a$) et $\Delta < 0$ (2 racines complexes conjuguées: $x_1 = -b/2a + j\sqrt{(}-\Delta)/2a)$ et $x_2 - b/2a - j\sqrt{(}-\Delta)/2a)$). Pour les racines complexes on affichera les parties réelles et imaginaires.

1.4 Calcul de moyenne

On désire écrire un programme dont le rôle est de calculer la moyenne de N notes rentrés au clavier. L'utilisateur doit fournir le nombre N de notes puis la liste des N notes à la suite de quoi le résultat (moyenne des N notes) doit s'afficher à l'écran. Le nombre maximal de notes est fixé à 100.

Que se passe t'il si l'utilisateur rentre d'abord la valeur 0? Proposer une solution pour tenir compte de ce cas.

Séance 2

(*boucle for, tableaux. fonctions.*)

2.1

2.1.1

Pour x variant entre x_{min} et x_{max} avec un pas de δx, calculer et afficher $y = ax^2 + bx + c$. Les coeficients réels a,b et c seront rentrés au clavier par l'utilisateur. On prendra par exemple $x_{min} = -1$ et $x_{max} = 1$

Le pas sera fixé par l'utilisateur. On affichera les valeurs de x et de y correspondantes.

2.1.2

Utilisez un tableau pour stocker les valeurs de y puis calculer à partir des valeurs du tableau les valeurs minimale, maximale et moyenne de y.

Créer des fonctions pour calculer chacune de ces valeurs (min, max. etc...).

2.1.3

Modifier la fonction calculant le minimum de y de manière à ce qu'elle renvoie aussi la valeur de x correspondant à ce minimum.

2.2

Un peu d'algorithmique: Ecrire un programme permettant de dessiner un sapin en fixant la hauteur du cône, la largeur et la hauteur du tronc et le nombre de boules (dont la position est aléatoire). Par exemple, pour un cone de hauteur 6, un tronc de hauteur 2 et de largeur 3 et 4 boules on doit obtenir un affichage du type:

```
     *
    *O*
   *****
  **O***O
 *****O***
**********
    |||
    |||
```

Pour résoudre le problème, il est judicieux de le décomposer en trois étapes:

- Dessiner le cône du sapin qui dépend de sa hauteur. Soit N la hauteur du cône, il s'agit d'imprimer à l'écran N lignes dont la première est composée d'une étoile et la dernière de 2*N-1 étoiles (voir dessin).
- Rajouter le tronc.
- Modifier le dessin du cône en rajoutant des boules de manière aléatoire.

Remarque: on peut générer un nombre aléatoire x compris entre 0 et 1 en écrivant:

```
x=rand()/RAND_MAX;
```

Suggestion pour l'étape 3: Avant d'afficher une étoile, générer un nombre aléatoire compris entre 0 et 1. Si ce nombre est inférieur à une certaine valeur (probabilité de placer une boule) et s'il reste des boules à placer, afficher un O sinon afficher une étoile.

Séance 3

(*boucle while, bibliothèque mathématique, lecture et écriture dans un fichier*)

3.1

On reprend le programme permettant de calculer la moyenne de notes saisis au clavier mais cette fois-ci l'utilisateur n'indique pas le nombre de notes avant de les rentrer mais termine la saisie en rentrant le nombre -1. En supposant que le nombre de notes ne dépassera jamais 30, écrire le programme permettant d'afficher la moyenne des notes ainsi que leur nombre.

3.2

On cherche à résoudre une équation du type $f(x) = 0$ par dichotomie.

On suppose connu un intervalle x_0, x_1 dans lequel la fonction f s'annule. On à donc $f(x_0).f(x_1) < 0$. On prend le point milieu du segment x_0,x_1: $x2 = (x0+x1)/2$ et on calcule $f(x_2)$. Si $f(x_2).f(x_0) > 0$ on sait que la racine se trouve sur le segment x_2, x_1 sinon elle est sur le segment x_0, x_2. Dans le premier cas on remplace x_0 par x_2, dans le second on remplace x_1 par x_2. On recalcule un point milieu, etc.. Ceci jusqu'a obtenir $|x1-x0| < \epsilon$, ϵ étant la précision, fixée à l'avance, que l'on souhaite.

Ecrire un programme appliquant cette méthode à l'équation $cos(x) = x$.

3.3

Lire des notes dans un fichier puis calculer et afficher le nombre de notes N, leur moyenne, la note minimale, la note maximale et l'écart type.

Remarque: l'écart type est défini par la relation:

$$\sigma = \sqrt{\sum_{i=0}^{N-1} (y_i - \overline{y})^2/(N-1)}$$

où $\overline{y}$ représente la moyenne des y_i.

Séance 4

(*switch, structures*)

4.1

Ecrire un programme qui affiche le nombre d'occurences de chaque voyelle (a, e, i, o, u, y) contenues dans le fichier `/home/aisem/texte.txt` (utiliser l'instruction *switch*).

4.2

Le fichier `/home/aisem/etudiants.dat` contient des couples: nom, numéro, le fichier `/home/aisem/notes.dat` contient des couples: numéro, note.

Ecrire un programme permettant de créer un fichier `resultats.dat` contenant les couples: nom, note à partir des deux fichiers précédents.

Algorithme suggéré:

- Définir un tableau de structures contenant 3 membres:
  - le nom (12 caractères)
  - le numéro (un entier)
  - la note (un entier)
- ouvrir le fichier *etudiants* et remplir les deux membres des éléments du tableau correspondant (nom et numéro). Conservez le nombre d'éléments du tableau (N).
- ouvrir le fichier *notes* puis pour chaque couple lu dans le fichier, chercher l'élément du tableau contenant le numéro lu et affecter la note correspondante.
- ouvrir le fichier *resultats* (en écriture) et écrire dans le fichier pour chaque élément du tableau le nom et la note correspondants (une ligne par couple).

Séance 5

L'objectif de ce TP est d'écrire un programme de jeu de cartes: la bataille!. Dans cette version, on jouera avec un jeu de 32 cartes. Après mélange et distribution des cartes a deux joueurs, chacun retourne la première carte de son tas. La carte la plus forte gagne la manche et les deux cartes sont mis en dernier dans le tas du gagnant. L'ordre est: As, Roi, Dame, Valet, 10, 9, 8, 7. En cas d'égalité, la couleur la plus forte est gagnante dans l'ordre: Coeur, Pique, Carreau, Trèfle. Le joueur qui ramasse toutes les cartes est déclaré gagnant.

Le programme fait appel a une structure pour représenter une carte dont les membres sont la valeur et la couleur. Les cartes de chaque joueur sont stockées dans un tableau.

Le squelette de programme suivant est fourni:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct carte
{
  int valeur;
  int couleur;
} Carte;

/*-----------------------------------------------------------------*/
void initjeu(Carte jeu[]) /* remplit le tableau avec les 32 cartes */
{}
/*-----------------------------------------------------------------*/
int alea32() /* genere un entier aleatoire compris entre 0 et 31 */
{
  return 0;
}
/*-----------------------------------------------------------------*/
void permutte(Carte *carte1, Carte *carte2) /* permutte 2 cartes */
{}
/*-----------------------------------------------------------------*/
void melange(Carte jeu[]) /* melange les cartes */
{}
/*-----------------------------------------------------------------*/
void affiche_carte(Carte X) /* affiche une carte (valeur,couleur)*/
{}
/*-----------------------------------------------------------------*/
void affiche(Carte jeu[],int N) /* affiche N cartes */
{}
/*-----------------------------------------------------------------*/
void distribution(Carte jeu[], Carte jeu1[], Carte jeu2[])
         /* distribue les cartes aux deux joueurs */
{}
/*-----------------------------------------------------------------*/
void jouer1coup( Carte jeu1[], Carte jeu2[], int*N1, int*N2)
         /* gere le resultat d'une bataille (1 coup) */
{}
/*-----------------------------------------------------------------*/
int main()
{
  Carte jeu[32];
  Carte jeu1[32]; /* cartes du joueur 1 */
  Carte jeu2[32];
  int N1=15; /* indice de la derniere carte joueur 1 */
  int N2=15;

  initjeu(jeu);
  srand(12500); /* initialisation du générateur aléatoire */
  melange(jeu);
  distribution(jeu,jeu1,jeu2);
```

```
  //  while () /* tant que la partie n'est pas terminee */
  {
        jouer1coup(jeu1,jeu2,&N1,&N2);
  }
  /* afficher qui gagne et en combien de coups */
  return 0;
}
```

Copier le dans votre répertoire en tapant la commande suivante dans la fenêtre terminal:

```
cp /home/aisem/squelette.c le_nom_que_vous_voulez.c
```

Il s'agit de le compléter en écrivant le code des fonctions définies.

5.1 Initialisation du jeu

Ecrire la fonction `initjeu` qui remplit le tableau `jeu` avec les 32 cartes ainsi que les fonctions d'affichage d'une carte et du jeu entier. Vérifier le fonctionnement avant de passer à la suite.

5.2 Mélange des cartes

Ecrire une fonction (`alea32`) qui renvoie un entier compris entre 0 et 31 (correspondant à un indice valide du tableau `jeu`). Ecrire une fonction permettant de permuter deux variables de type `Carte`. Utiliser ces deux fonctions pour écrire la fonction de mélange des cartes. L'idée est de réaliser un nombre important (50 par exemple) de permutations entre deux cartes tirées au hasard. Vérifier en l'affichant que le jeu est bien mélangé.

5.3 Distribution

Ecrire la procédure permettant de distribuer 16 cartes à chaque joueur. On utilisera 1 tableau de 32 `Carte`s pour chaque joueur. Afficher les 16 cartes de chacun des joueurs et vérifier le bon fonctionnement.

5.4 Jeu de la carte

Voici la partie la plus délicate du programme.

Chaque joueur possède N1 (ou N2) cartes indicées de 0 à N1-1 (ou N2-1). La carte jouée est celle d'indice 0. Une solution consiste à mettre de coté les cartes d'indice 0 de chaque joueur puis à décaler toutes les cartes restantes de chaque joueur vers le bas (cartes 1 à N1-1 (ou N2-1) mises de 0 à N1-2 (ou N2-2)).

Les deux cartes mises de coté sont alors comparées (valeur et éventuellement couleur) et rangées dans le tas du gagnant aux positions N1-1 (pour la carte gagnante) et N1 si le joueur 1 gagne ou N2-1 et N2 si c'est le joueur 2.

Les valeurs de N1 et N2 doivent bien sur être ajustées en conséquence (+1 pour le gagnant, -1 pour le perdant).

Tester le bon fonctionnement sur quelques coups du jeu en affichant par exemple les cartes restantes des deux joueurs après chaque coup.

5.5 Conclusion

Compléter le programme principal en écrivant la condition de fin de partie. Afficher le joueur gagnant et le nombre de coups joués. On pourra aussi afficher la succession des coups joués.

Séance 6

et 7

Les deux dernières séances seront consacrées aux thèmes vues en TD. On rappelle simplement les objectifs des programmes à écrire.

6.1 Droite des moindres carrés

La droite des moindres carrés est une droite qui passe au mieux à travers un ensemble de points x_i, y_i. L'équation de cette droite est donnée par: $y = ax + b$ avec:

$$a = \frac{\overline{xy} - \overline{x}.\overline{y}}{\overline{x^2} - \overline{x}^2} \qquad b = \frac{\overline{y}.\overline{x^2} - \overline{x}.\overline{xy}}{\overline{x^2} - \overline{x}^2}$$

L'objectif est d'écrire un programme permettant de calculer et d'afficher la valeur de $y = ax + b$ en fonction d'une valeur de x rentrée au clavier.

On suppose que les points x_i, y_i sont contenus dans un fichier et sont au maximum au nombre de 30.

6.2 Annuaire

L'exercice consiste à concevoir un logiciel de gestion d'un annuaire basé sur un tableau de structures.

Les données de l'annuaire sont stockées dans un fichier.

Les services offerts doivent permettre de:

- Lire le fichier et stocker les données dans un tableau
- Afficher les fiches
- Afficher la fiche contenant un nom rentré au clavier
- Ajouter une fiche dans la base
- Modifier une fiche
- Supprimer une fiche (et réorganiser le tableau!)
- Sauver les données dans le fichier.

Tous ces services doivent être accessibles à partir d'un menu.