IUT — DUT 1re Année
Université de Marne la Vallée — Année 2012-2013

Prog UNIX : Programmation modulaire
TP n°3

Dans tous les exercices, on veillera particulièrement à la lisibilité du code. Le but de ce TP est de modulariser un code en vrac (pas écrit par vous même) pour en faire un projet bien structuré. Dans un second temps, et pour les plus rapides, on veillera à améliorer le code.

Exercice 1 Modularisation d'un projet

Téléchargez l'archive `code_en_vrac.tar.gz` situé à l'adresse suivante :
http://www-igm.univ-mlv.fr/~borie/cours/prog_sys/code_en_vrac.tar.gz
Cette archive contient un fichier de code écrit en langage C : `code_en_vrac.c`. Il y a aussi un répertoire `data` avec un fichier `repertoire.txt` dedans.

Vous pouvez tenter de vérifier le bon (ou mauvais) fonctionnement de ce code en le compilant avec la commande `gcc -o test code_en_vrac.c -Wall -ansi` puis en exécutant le programme généré. L'exercice consiste maintenant à diviser ce code en plusieurs modules pour en faciliter la manipulation. Ce code résout en partie ce qui vous a été demandé dans le TP n°2 : gérer un répertoire de personnes.

Parmi les difficultés de cet exercice :

- Gérer le découpage en module.
- Faire les fichiers d'en-tête proprement (utilisez le cours!).
- Inclure les bibliothèques standards `stdio.h` et `string.h` dans les fichiers de code que lorsque c'est nécessaire. ÉVITEZ, autant que possible, l'inclusion des bibliothèques dans les fichiers d'en-tête.
- Faire un Makefile adapté au projet ainsi divisé.

Exercice 2 Un `main` facile à utiliser

Une fois bien divisé, votre projet devrait comporter un fichier `main.c` contenant une fonction `main` (c'est d'ailleurs le seul fichier à contenir une telle fonction).

L'objectif de cet exercice est de réécrire la fonction `main` pour faciliter l'utilisation de l'exécutable généré. Modifiez donc cette fonction `main` pour qu'elle donne l'affichage suivant:

```
Bienvenu dans le projet repertoire, faites un choix :
1 - Charger le repertoire depuis le fichier
2 - Afficher le repertoire courant
3 - Ajouter une fiche dans le repertoire courant
4 - Sauvegarder le repertoire courant
5 - Quitter
Votre choix :
?
```

Ensuite le code de votre fonction `main` devra faire le bon aiguillage suivant la fonctionnalité demandée par l'utilisateur.

Utiliser un `switch ... case` *est une bonne manière de faire.*

Exercice 3 **Fonctionnalités supplémentaires**

Vous pouvez ajouter les fonctions suivantes à chaque fois dans le module approprié :

- Une fonction `int compare_date(Date *d1, Date *d2)` qui retourne 1 si la date `*d1` est antérieure à la date `*d2`, -1 si la date `*d1` est postérieure à la date `*d2` et 0 si les deux dates sont égales.
- Une fonction `Fiche* recherche_fiche(Repertoire *R, char *mon)` qui recherche si le répertoire `*R` contient une fiche dont le nom de la personne est `*nom`. Si aucune fiche est trouvée, la fonction retourne un pointeur `NULL` sinon elle retourne un pointeur vers la première fiche trouvée.
- Une fonction `void affiche_personne_mois(Repertoire *R, int mois)` qui affiche les fiches du répertoire `*R` où la personne correspondante est née durant le mois dont le numéro est donné en argument `mois`.
- Une fonction `void affiche_plus_jeune(Repertoire *R)` qui demande à l'utilisateur de saisir une date au clavier et affiche ensuite toutes les personnes nées après la date donnée au clavier présente dans le répertoire `*R`.

Exercice 4 **Finalisation du rendu**

Voici le cahier des charges pour ce second TP :

- votre travail devra être rendu en pièce attachée d'un mail envoyé à une adresse communiquée par votre chargé de TP,
- vous ne devrez envoyer qu'une seule pièce jointe au format `tar.gz` dont le nom devra être : `TP1_NOM_PRENOM.tar.gz` où `NOM` est votre nom de famille et `PRENOM` votre prénom (voir l'aide mémoire `tar` en ligne),
- le sujet de votre mail devra être : `[DUT1 info][TP2 prog_sys] NOM PRENOM` où `NOM` devra être remplacé par votre nom de famille et `PRENOM` par votre prénom,
- votre rendu devra contenir au moins trois répertoires aux noms de `doc` ,`sources`, `data` et un fichier nommé `Makefile`,
- une fois votre TP décompressé, la commande `make` devra compiler les sources C des exercices situées dans le dossier `sources` et fabriquer un exécutable au nom adapté qui se placera dans la racine de votre TP. La compilation devra être effectuée avec les options `-Wall` et `-ansi`,
- L'exécutable de votre rendu devra tester, autant que possible, toutes les fonctions implantées dans vos sources.
- la commande `make clean` devra effacer tous les exécutables générés lors de l'exécution de la commande `make`.