École Centrale de Nantes **2012-2013**

Algorithmique et Programmation

TP1
Initiation Linux et C
Structures algorithmiques de base et traduction C

Lorsque vous écrivez du code ou que vous utilisez le terminal, sachez qu'une documentation abondante est présente en ligne, entre autres sur le serveur pédagogique. Donc avant de poser une question aux chargés de TP : **LISEZ LA DOCUMENTATION !**

FICHIERS À RENDRE SUR MARKUS :
- `helloworld.c`
- `ADeboguer.c`
- `mon1erprog.c`
- `rapport_TP1.pdf`

1 Linux et terminal

1.1 Fenêtre de commandes

Connectez-vous sous `linux` avec votre nom d'utilisateur et votre mot de passe. Ouvrez une fenêtre de commande (parfois aussi appelée Terminal, Console ou autre) avec l'icône correspondant à *Menu Principal / Système / Terminaux / Konsole* à partir de l'angle gauche de la barre de tâches.

Testez des commandes `linux` que vous trouverez dans les documents en ligne sur le serveur pédagogique https ://pedagogie.ec-nantes.fr/spip/ et en particulier les commandes :	`ls`
	`pwd`
	`man gcc`

Pour quitter une page de manuel ouverte avec `man` :	`q`

La plupart des commandes sont mnémotechniques, par exemple la commande `cd` signifie change directory, la commande `ls` list, la commande `mkdir` make directory, etc.

1.2 Création et utilisation de vos répertoires

Créez dans votre répertoire personnel un répertoire de nom `algpr`, fils du répertoire courant, pour ranger le travail des TP ALGPR :	`mkdir algpr`
Descendez dans ce répertoire :	`cd algpr`
Créez à nouveau un répertoire, fils du répertoire `algpr`, que vous nommerez par exemple `tp1` ou `tp_init`, pour ranger les fichiers de ce premier TP d'initiation :	`mkdir tp1`
Descendez dans tp1 :	`cd tp1`
Pour remonter dans algpr :	`cd ..`
Pour savoir où vous êtes :	`pwd`

Consultez les documents en ligne sur Linux pour approfondir et maitriser la gestion de votre arborescence, en particulier pour savoir déplacer ou renommer `mv` (move), copier `cp` (copy) ou détruire des fichiers `rm` (remove).

1.3 Manipulation terminal

Auto-complétion : la touche `tabulation` ou TAB vous permet de compléter automatiquement les noms de commandes et de fichiers que vous tapez (1x si une solution, 2x si plusieurs solutions).
Historique des commandes : les touches HAUT et BAS vous permettent de naviguer dans l'historique des commandes du terminal pour rappeler une commande.
Modification d'une commande : les touches DROITE et GAUCHE vous permettent de déplacer le curseur dans la commande pour modifier cette dernière.

2 Initiation C

2.1 Écriture de code source – "Hello world !"

Objectif : Écrire le code source du programme qui affiche "Hello world !" dans le terminal.
Étapes :
- Ouvrez un **éditeur de code** (Jedit, gedit ou nedit, mais pas office),
  Remarque : Si vous ouvrez votre éditeur en ligne de commande, le caractère `&` garde active la fenêtre de commande en même temps que la fenêtre de l'éditeur, par exemple : `jedit &`
- Sauvegardez votre fichier avec un nom explicite et l'extension `.c`, par exemple `helloworld.c`, dans votre répertoire `tp1`, ceci vous permet de bénéficier de la coloration syntaxique,
  Remarque : Avec l'extension `.c`, l'éditeur reconnait un programme `C`.
- Écrivez l'en-tête du fichier correctement,
- Faites les inclusions nécessaires : `stdlib` et `stdio`,
- Écrivez le squelette de la fonction principale `main`,
- Écrivez l'instruction permettant d'afficher "Hello World !", avec la commande `printf`.

2.2 Compilation, Débogage et Exécution de code

Vous avez peut-être quelques erreurs dans votre programme. Vérifiez si la compilation et l'exécution se déroulent correctement.
- Dans un terminal, déplacez-vous jusqu'au répertoire qui contient votre fichier : commandes `cd`, avec les commandes `ls` et `pwd` pour vous aider à situer le fichier,
- Lancez la compilation complète (compilation et édition de liens) avec la commande `gcc` :
  `gcc` *`nomFichier`*`.c -o` *`nomExecutable`*`.out`
- Corrigez les erreurs au besoin. Une erreur de compilation est signalée avec le nom de la fonction, le numéro de la ligne et une indication de l'erreur :
  ```
  helloWorld.c : In function ‘main’ :
  helloWorld.c :6 : error : expected ‘ ;’ before ‘return’
  ```
  Votre éditeur de code doit vous indiquer la ligne, option par défaut ou à activer selon l'éditeur.
- Lancez l'exécution de votre programme dans le terminal,
  `./nomExecutable``.out`
  Le `./` est nécessaire pour indiquer au terminal que vous exécutez un programme dans le dossier courant.
  Si "Hello World !" s'affiche vous pouvez continuer le TP,
- Si vous avez des erreurs d'exécution, reprenez votre code ligne par ligne pour les détecter.

2.3 Compilation et édition de liens

Nous allons faire maintenant la distinction entre l'étape de compilation proprement dite et l'étape d'édition de liens. Cette distinction sera utile lorsque nous aurons plusieurs fichiers sources à compiler pour produire un exécutable. Vous pouvez utiliser un des deux programme précédent.

– **Compilation seule**, qui produit un fichier objet `.o` à partir d'un code source :
`gcc -c nomFichier.c`
– Vérifiez le contenu de votre dossier de travail, il doit contenir un fichier *nomFichier*`.o`
– **Édition de liens**, qui produit l'exécutable à partir d'un fichier objet (ou plusieurs fichiers objets comme nous le verrons dans les TP suivants) :
`gcc -o nomExecutable.out nomFichierObjet.o`

2.4 Analyse de code source – Débogage

Téléchargez et compilez le ficher `ADeboguer.c`, disponible sur le serveur pédagogique, pour corriger toutes les erreurs.
Il est conseillé de corriger les erreurs une par une, **en commençant par la première de la liste**, car les erreurs peuvent avoir des effets de bord et produire d'autre erreurs.
Lorsque vous avez corrigé toutes les erreurs de compilation. Exécutez le programme produit, testez le avec plusieurs valeurs [-1, 0, 1] et corrigez au besoin des erreurs d'exécution.

3 Modification de code source

Vous trouverez ci-dessous un algorithme pour calculer n termes de la suite de Fibonacci. Le programme est déjà écrit, dans le fichier `mon1erprog.c`. Il est disponible sur le serveur pédagogique de l'école.

problème : Le programme mon1erprog permet de calculer n termes de la suite de Fibonacci (suites définies par récurrence $u_{n+1} = u_n + u_{n-1}$) obtenue à partir de $u_0 = 1$ et $u_1 = r$ avec $r = (1-\sqrt{5})/2$. La valeur r est telle que $r \in]-1, 0[$ et $r^2 = r + 1$. On peut montrer (par récurrence) que pour tout n, on a $u_n = r^n$, d'où la convergence théorique de la suite (u_n) vers 0. Le but du programme est de mettre en évidence l'effet de la propagation des erreurs d'arrondi dans le calcul par récurrence de la suite (u_n).

```
constante
        réel r

algorithme
variables
        entier n
        réels uprec, ucour, usuiv / pour u_{i-2}, u_{i-1}, u_i/
début
        / initialisations /
        n ← lire( ) /lecture de n au clavier/
        uprec ← 1
        ucour ← r
        / calcul de u_2, ..., u_n /
        pour i ← 2 à n faire
                usuiv ← ucour + uprec
                uprec ← ucour
                ucour ← usuiv
                / affichage de u_i calculé par récurrence, et de u_i calculé directement par sa valeur r^i /
                écrire(i,ucour,r^i)
        fin pour
fin
```

3.1 Téléchargement et compilation

– Enregistrez une copie de ce fichier sur votre compte, dans le répertoire approprié `tp1`,
– Vérifiez que le fichier est bien dans votre répertoire `tp1`, en utilisant la commande `ls`,
– Visualisez le contenu du fichier avec la commande `cat` : `cat mon1erprog.c`,
– Compilez le fichier pour créer un exécutable `./mon1erprog.out`. N'oubliez pas d'ajouter `-lm` dans la commande d'édition de liens pour la librairie mathématique,
– Lancez l'exécution : `./mon1erprog.out` (Attention à la lecture au clavier !).

3.2 Modifier le programme

– Éditez le code source,
– **Faites les modifications indiquées en commentaire dans le code source,**
– Sauvegardez et lancez la compilation,
– Déboguez au besoin (modification et compilation),

3.3 Analyse des résultats

- jeux d'essais

Exécutez le programme pour plusieurs valeurs de n. Retenez une valeur de n pour laquelle l'observation des résultats est intéressante. Exécutez alors le programme pour cette valeur avec la commande : `./mon1erprog.out >out.txt`

Le flux de sortie est alors redirigé vers le fichier `out.txt` : chaque affichage à l'écran devient une écriture dans le fichier `out.txt`. Vous obtenez ainsi un fichier contenant les résultats obtenus.

- analyse des résultats

Le programme calcule bien des valeurs de r^n qui s'approchent de 0. En revanche, les valeurs de u_n calculées par récurrence divergent assez rapidement. Cela vient de la propagation de l'erreur d'arrondi ε faite par l'ordinateur dans le calcul de r. En effet, si on note $\tilde{u}_n$ les valeurs calculées, on obtient : $\tilde{u}_1 = u_1 + \varepsilon$, $\tilde{u}_2 = u_2 + 2\varepsilon$, $\tilde{u}_3 = u_3 + 3\varepsilon$, $\tilde{u}_4 = u_4 + 5\varepsilon$, $\tilde{u}_5 = u_5 + 8\varepsilon$, $\tilde{u}_6 = u_6 + 13\varepsilon$, ... $\tilde{u}_n = u_n + a_n\varepsilon$,... avec a_n qui tend vers l'infini.

– Commentez les résultats obtenus dans un rapport en PDF que vous rendrez avec vos fichiers sources.

3.4 Pour aller plus loin : Automatisation de la compilation

Cette section est optionnelle.
Vous avez peut-être remarqué à quel point il est fastidieux de rappeler les mêmes commandes de compilation, surtout lors du débogage. Nous allons donc automatiser le processus de compilation avec la commande `make` et le fichier `makefile`, dans lequel nous allons écrire des règles :

```
#commentaire (optionnel)
nomCible : listes des fichiers nécessaires
[tabulation] commande
```

Attention ! Il n'y a qu'un seul fichier `makefile` par répertoire. Ce fichier peut contenir des règles pour produire différents programmes.

- Supprimez les fichiers `helloWorld.o` et `helloWorld.out`
- Dans votre éditeur préféré, créez un nouveau fichier et sauvegardez le avec le nom `makefile`, **sans extension**,
- La première règle du fichier `makefile` est exécutée par défaut. Par convention, nous nommons cette règle `all` et nous indiquons les exécutables produits par le `makefile` en liste de fichiers nécessaires, il n'y pas de *commande* dans cette règle :

  ```
  # regle par defaut
  all : nomExecutable1.out nomExcutable2.out etc...
  ```

  Ajoutez une règle par défaut pour produire l'exécutable `helloWorld.out`,
- Ajoutez une règle de compilation seule pour le code source `helloWorld.c`,
- Ajoutez une règle d'édition de liens pour créer l'exécutable `helloWorld.out` à partir du fichier objet `helloWorld.o`.
- Exécutez la commande `make` dans le terminal. Cette commande doit produire les fichiers `helloWorld.o` et `helloWorld.out` de manière automatique.

- Ajoutez ou modifiez les règles pour produire aussi l'exécutable `ADeboguer.out`.

Avec la commande `make`, il est possible de spécifier la règle que l'on souhaite exécuter. Pour illustrer cette possibilité :
- Ajoutez une règle nommée `clean` qui n'a besoin d'aucun fichier et dont la commande supprime tous les fichiers objets et les exécutables du dossier de travail. Vous pourrez appeler cette règle pour nettoyer votre dossier avec la commande `make clean`.