IFT1169 – TRAVAIL PRATIQUE #3 – 07 mars 2011

« 53 2B 2B[1] »

Mohamed Lokbani

Équipes : le travail peut-être fait en binôme mais vous ne remettez qu'un travail par équipe.
Remise : une seule remise est à effectuer par voie électronique **le lundi 28 mars 2011, 23h59 au plus tard, sans possibilités de prorogation**.
Conseils : n'attendez pas le dernier jour avant la remise pour engager votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : ce TP a pour but de vous faire pratiquer les différentes notions en rapport avec les entrées et les sorties.

Énoncé : ce travail est composé de 2 exercices. Les étapes de compilation et d'édition de liens pour les 2 exercices doivent être regroupées dans un seul fichier « makefile ». Chaque cible dans le fichier correspond à un exercice.

Exercice	Cible
1	exo01
2	exo02

Il est permis d'utiliser pour ce travail le type « string » tel que décrit dans le fichier « string911.cpp » disponible sur la page web de cet annoncé. Par contre, vous ne pouvez pas vous servir des conteneurs de la STL.

Exercice -1-
Il s'agit d'écrire un convertisseur hexadécimal d'un fichier fourni en entrée et de préserver le résultat de cette conversion dans un fichier en sortie en respectant un format approprié décrit dans l'annexe -1-. Le fichier en entrée est un fichier texte (ascii). L'appel d'un tel programme se fait comme suit :

./exo01 test.txt

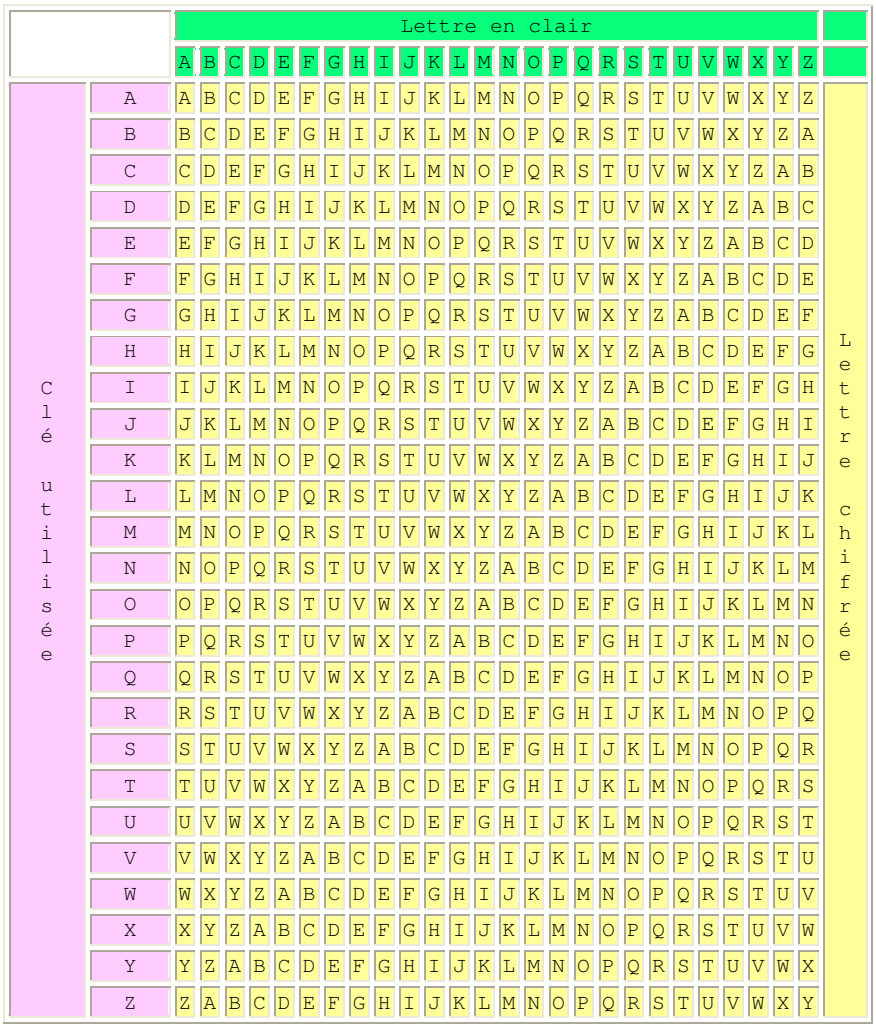
Le programme va se charger de nommer lui-même le fichier résultat à partir du nom du fichier fourni en entrée, en y ajoutant « hex » comme suit : « hex_test.txt ».

Exercice -2-
Blaise de Vigenère diplomate français de la cour d'Henry III de France, (1523-1596), publia en 1586, dans son Traité des chiffres le principe du chiffre carré connu plus tard sous le nom du chiffrement de Vigenère. Ce chiffrement se base sur la table de Jean Trithème (il l'a nommée tabula recta). Dans cette table, l'alphabet est répété sur 26 lignes avec un décalage à gauche d'une lettre pour chaque nouvelle rangée. Il est amélioré en 1553 par Giovan Batista Belaso. Ce dernier a introduit la notion de clef qu'il appelle mot de passe. En 1585, Vigenère améliore la table de Trithème en modifiant la façon d'utiliser la clef. Cette technique a introduit une amélioration décisive de la méthode de César qui était trop facile à craquer.

Carré de Vigenère

La méthode de Vigenère consiste à substituer à chaque lettre du message à crypter, une lettre de l'alphabet calculée à partir d'une clé. Le calcul de cette clé est basé sur le Carré de Vigenère comme suit:

[1] Après cryptage avec la clé « Programmation » et conversion du résultat en hexadécimal.



Mot	D	e	u	x		e	q	u	i	p	e	s		d	e		f	o	o	t	b	a	l	l
Clé	P	r	o	g		r	a	m	m	a	t	i		o	n		P	r	o	g	r	a	m	m
Rés.	S	v	i	d		v	q	g	u	p	x	a		r	r		u	f	c	z	s	a	x	x

Nous répétons autant de fois la clé de cryptage sur la phrase à crypter, uniquement sur les caractères alphabétiques. Par la suite, nous cherchons dans le carré de Vigenère la lettre cryptée (l_i) qui correspond à la paire (m_i, c_j) où i représente la ième lettre dans la phrase à crypter et j la jième lettre dans la clé de cryptage.

Vous trouverez plus de détails sur cette technique à la page web :

http://fr.wikipedia.org/wiki/Chiffre_de_Vigen%C3%A8re

Il vous est demandé de réaliser le programme nécessaire pour crypter et décrypter un fichier donné, en tenant compte des options suivantes :

exo02 -h	L'option « -h » permet d'afficher une description des commandes nécessaires pour utiliser votre programme
exo02 –c clé fichier.txt	L'option « -c » permet de spécifier à votre programme que l'opération à réaliser est « cryptage ». Le premier argument est la « clé » nécessaire pour crypter le fichier « fichier.txt ». Le résultat du cryptage est préservé dans le fichier « fichier_sec.txt »
exo02 –d clé fichier_sec.txt	L'option « -d » permet de spécifier à votre programme que l'opération à réaliser est « décryptage ». Le premier argument est la « clé » nécessaire pour décrypter le fichier « fichier_sec.txt ». Le résultat du décryptage est préservé dans le fichier « fichier.txt »

Fichiers à remettre : vous devez remettre les fichiers nécessaires pour une compilation et une exécution correctes de vos programmes. Vous devez inclure aussi un fichier « makefile » permettant d'automatiser la compilation de vos fichiers. Par ailleurs, vous devez inclure dans votre remise un rapport décrivant le travail effectué. Ce rapport doit porter le nom « tp3H11.pdf ».

Remise : il est important de noter que votre TP sera compilé avec « gcc4.5.0 ». Si par choix vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI. Pour avoir la version du compilateur, utilisez la commande "**gcc –v**" qui devra donner le numéro de version « **4.5.0** ».

Par ailleurs, assurez-vous de la présence de l'option « pedantic » sur la ligne de compilation.

Pour éviter d'omettre un des fichiers associés à ce travail lors de la remise, nous vous conseillons de regrouper l'ensemble des fichiers dans un seul fichier compressé **tp3h11.zip** et de remettre ce dernier comme suit :

1. commencez d'abord par vous connecter sur la machine « **remise** » comme cela a été pratiqué dans la démo #01.
2. envoyez l'ensemble de vos fichiers par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un Xterm : man remise). Respectez les noms des fichiers.

   remise ift1169 1169tp3 tp3h11.zip

3. vérifiez que la remise s'est effectuée correctement.

   remise –v ift1169 1169tp3

À noter que si le serveur vous refuse la connexion ssh, utilisez celui-ci : frontal07.iro.umontreal.ca

Automatisation de la correction : nous allons utiliser des scripts pour compiler, exécuter et corriger votre programme. Assurez-vous de respecter les noms de fichiers.

Pour comparer votre sortie avec celle fournie en exemple et si la sortie n'est pas un fichier (redirection directe), redirigez votre sortie dans un fichier avec le symbole « > » comme suit : « tp3.exe > masortie.txt ». Par la suite, comparez les deux sorties (obtenue et fournie) en vous aidant pour cela de la commande « diff » sous « MinGW/MSys » :

```
diff –b –w –i –B sortiefournie.txt votresortie.txt
```

Sous « DOS », utilisez plutôt la commande « fc » tout en ajustant ses options. Si les fichiers sont identiques, vous n'obtiendrez rien en sortie. Sinon, vous obtiendrez les différences entre les deux fichiers. Pour plus de détails sur cette commande « unix », sur votre console « xterm » exécutez la commande suivante: « man diff ».

Si la différence est provoquée par des caractères non imprimables (« bizarroïdes »), il faudra vous assurer que le fichier texte est dans le bon format d'encodage. Pour cela, utilisez par exemple :

« Conversion dos à unix » : **dos2unix < entree > sortie**

« Conversion unix à dos » : **unix2dos < entree > sortie**

Barème : ce TP3 est noté sur **14 points**. Les points sont répartis comme suit :

Exercice 1	**6**
Exercice 2	**8**

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ….

- La non remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0.
- Un programme qui compile mais ne réalise pas les choses prévues dans la spécification : 0.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Le non respect du nom du fichier va générer une erreur de compilation donc un des points de la spécification n'a pas été respecté : 0.
- Aberration dans le codage : même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Des questions à propos de ce TP?

Une seule adresse : dift1169@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1169] et une référence au **tp03**.

Mise à jour

06-03-2011 diffusion de l'énoncé.

Annexe -1-

Soit le fichier « 1test.txt », dont le contenu est comme suit :

```
Deux equipes de football, composees chacune de onze vaches portant dossard
numerote et trottinant sur un faux pre de couleur orange situe le long de
l'autoroute entre Amsterdam et Utrecht, ont provoque un embouteillage de plus de
douze kilometres mardi.
```

La commande « exo01 1test.txt » génère la sortie suivante, préservée dans le fichier « hex_1test.txt » :

```
Conversion hexadecimale du fichier: '1test.txt'

00000000:  44 65 75 78 20 65 71 75 | 69 70 65 73 20 64 65 20  Deux equipes de
00000010:  66 6F 6F 74 62 61 6C 6C | 2C 20 63 6F 6D 70 6F 73  football, compos
00000020:  65 65 73 20 63 68 61 63 | 75 6E 65 20 64 65 20 6F  ees chacune de o
00000030:  6E 7A 65 20 76 61 63 68 | 65 73 20 70 6F 72 74 61  nze vaches porta
00000040:  6E 74 20 64 6F 73 73 61 | 72 64 20 6E 75 6D 65 72  nt dossard numer
00000050:  6F 74 65 20 65 74 20 74 | 72 6F 74 74 69 6E 61 6E  ote et trottinan
00000060:  74 20 73 75 72 20 75 6E | 20 66 61 75 78 20 70 72  t sur un faux pr
00000070:  65 20 64 65 20 63 6F 75 | 6C 65 75 72 20 6F 72 61  e de couleur ora
00000080:  6E 67 65 20 73 69 74 75 | 65 20 6C 65 20 6C 6F 6E  nge situe le lon
00000090:  67 20 64 65 20 6C 27 61 | 75 74 6F 72 6F 75 74 65  g de l'autoroute
000000A0:  20 65 6E 74 72 65 20 41 | 6D 73 74 65 72 64 61 6D   entre Amsterdam
000000B0:  20 65 74 20 55 74 72 65 | 63 68 74 2C 20 6F 6E 74   et Utrecht, ont
000000C0:  20 70 72 6F 76 6F 71 75 | 65 20 75 6E 20 65 6D 62   provoque un emb
000000D0:  6F 75 74 65 69 6C 6C 61 | 67 65 20 64 65 20 70 6C  outeillage de pl
000000E0:  75 73 20 64 65 20 64 6F | 75 7A 65 20 6B 69 6C 6F  us de douze kilo
000000F0:  6D 65 74 72 65 73 20 6D | 61 72 64 69 2E 0A         metres mardi..
```

Nous décomposons les éléments par segment de 16 octets chacun où chaque octet représente un caractère. Chaque segment a son adresse de départ, affichée dans un format hexadécimal (00000000, 00000010 etc.). Cette adresse est suivie de deux groupes de valeurs séparées par le caractère « | ». Chaque groupe contient 8 valeurs hexadécimales (8 octets).

Ainsi donc la lettre « D » est transformée en « 44 », après conversion en ascii en base 16 (hexadécimal). La conversion des 16 premiers caractères se fait comme suit :

D	e	u	x		e	q	u	i	p	e	s		d	e	
44	65	75	78	20	65	71	75	69	70	65	73	20	64	65	20

Finalement la dernière colonne contient les 16 caractères. Pour cet exercice, on se contente de représenter les caractères dont le code ascii est compris entre 32 et 126. Tout caractère en dehors de ce segment autorisé, est transformé en un point. Dans le précédent exemple, le dernier caractère du texte est un saut de ligne « 0A » (10 en base 10). Il a été transformé en un point dans l'affichage final.

Code Ascii
http://fr.wikipedia.org/wiki/Ascii

Système hexadécimal
http://fr.wikipedia.org/wiki/Syst%C3%A8me_hexad%C3%A9cimal