



EXAMEN DE LABORATOIRE ASSEMBLEUR 2^{ème} SESSION 2006

NOM :
Prénom :
Groupe :
Professeur : BEJ - HAL - JCJ – NVS - PBT

L'examen se déroule comme suit :

- Vous disposez de quatre (4) heures pour cet examen.
- Vous travaillez sur les machines de l'école (pas de portable).
- Vous avez accès au contenu de vos répertoires de travail sous MS-DOS et LINUX.
- Vous pouvez utiliser vos notes de cours.
- Veillez à taper du code lisible. Pour vous y aider, nous imposons la présence de commentaires (rôle des variables, brève description de chaque fonction).
- Déposez régulièrement vos sources dans le eCasier de votre professeur. Rassemblez vos sources dans un dossier portant votre nom et suffixé par V1, V2, etc.

Bon travail !

PREMIÈRE PARTIE : PARTIE ASSEMBLEUR SOUS LINUX

1. Directives générales

Toutes les fonctions qui vous sont demandées reçoivent tous leurs arguments par la pile. Aucune d'elles ne nettoie la pile, c'est toujours la procédure appelante qui s'en charge.

Aucune variable globale ne peut être utilisée, sauf celles explicitement autorisées (voir plus bas).

Vérifier le bon fonctionnement de la fonction demandée avant de passer à la fonction suivante.

2. Description détaillée

2.1) Écrivez une fonction « `strlen` » qui reçoit en argument l'adresse du premier caractère d'une chaîne de caractères terminée par un zéro binaire. La fonction ne modifie pas cette chaîne de caractères. Elle retourne dans `eax` la valeur binaire égale à la longueur de la chaîne, le zéro binaire final exclu.

Ainsi, par exemple, si la chaîne fournie en argument de `strlen` est :

"bonjour !",0

la valeur 9 est stockée dans `eax` au retour de `strlen`.

2.2) Écrivez une fonction « `afficher` » qui affiche à l'écran la chaîne de caractères terminée par un zéro binaire qu'elle reçoit en argument. Cette chaîne n'est pas modifiée. Aucun retour n'est attendu.

2.3) Écrivez une fonction « `estPalindrome` » qui stocke dans le registre `eax` la valeur zéro (0) ou un (1) selon que la chaîne de caractères terminée par un zéro binaire qu'elle reçoit en argument n'est pas ou est un palindrome.

Pour rappel, un palindrome est un mot qui se lit de la même manière de gauche à droite et de droite à gauche, comme le mot *ressasser* ou le nombre *12321*.

2.4) Utiliser les trois fonctions dont le développement vous est demandé ci-dessus au sein d'une fonction principale « `main` » telle que le programme correspondant, « `palindrome` », ait le comportement suivant. « `palindrome` » reçoit au moins un argument par la ligne de commande. Il produit en sortie, pour chaque argument facultatif fourni, un affichage renseignant si cet argument est un palindrome ou non.

De manière exhaustive :

*) si le nombre d'arguments facultatifs fournis par la ligne de commande est nul (égal à zéro), la chaîne de caractères :

```
"Usage : ./palindrome mot1 [mot2...moti]"
```

suivie d'un passage à la ligne, est affichée puis le programme retourne.

Cette chaîne de caractères, avec passage à la ligne final, peut être déclarée comme variable globale.

Ainsi par exemple, si l'utilisateur encode :

```
'./palindrome'
```

la chaîne :

```
"Usage : ./palindrome mot1 [mot2...moti]"
```

est affichée ;

*) si le nombre d'arguments fournis par la ligne de commande est d'au moins un (1), pour chaque argument, une ligne indiquant s'il s'agit d'un palindrome ou non est affichée.

Ainsi par exemple, si l'utilisateur encode :

```
'./palindrome 1234321 112233 1'
```

les chaînes :

```
"1234321 est un palindrome"
```

```
"112233 n'est pas un palindrome"
```

```
"1 est un palindrome"
```

sont affichées. Les chaînes :

```
" n'est pas un palindrome"
```

et :

```
" est un palindrome",
```

avec passage à la ligne final, peuvent être déclarées comme variables globales.

DEUXIÈME PARTIE : PARTIE ASSEMBLEUR SOUS MS-DOS

Introduction :

Dans cette exercice, on vous demande d'écrire un programme assembleur permettant de réaliser une opération arithmétique se trouvant dans un fichier et d'afficher le résultat à l'écran.

Description détaillée

On met à votre disposition dans **eDistri\hal\examen**, 4 fichiers textes nommés respectivement : plus.txt, fois.txt, moins.txt, divise.txt. Chacun des fichiers comporte une chaîne de maximum 20 caractères représentant deux opérandes séparés par un opérateur arithmétique et se terminant par le caractère égal (=).

Exemple du contenu du fichier moins.txt : 1234-543=

Notez que toutes les valeurs numériques, qu'il s'agisse de données ou de résultats doivent être considérées comme des entiers non signés.

Vous devez faire un menu assez simple afin que l'utilisateur puisse choisir l'opération qu'il souhaite effectuer (le fichier sur lequel il va travailler) ainsi que la possibilité de quitter. Pour cela, on vous propose de réaliser le menu suivant:

Taper 1 pour l'addition, 2 pour soustraction, 3 pour la multiplication, 4 pour la division et Q pour quitter.

NB : votre application doit permettre à l'utilisateur d'effectuer plusieurs opérations avant de quitter

Exigence de programmation

Dans cet examen, on vous impose d'utiliser un minimum de variables globales. En fait, à part les variables de type chaîne vous devez utiliser deux autres variables nommées nb1 et nb2, servant à gérer les opérandes. Toutes les autres données sont soit des variables locales, soit des constantes.

Vous devez écrire une fonction CALCUL, permettant de faire l'opération demandée. Cette dernière doit recevoir tous les arguments par la pile mais ne s'occupe pas de son nettoyage. En effet c'est toujours la procédure appelante qui s'en occupe.

L'affichage

Vous travaillez dans un environnement MS-DOS en mode d'affichage texte (80 colonnes, 25 lignes).

Vous devez afficher un premier écran contenant votre nom, prénom et groupe.
Un deuxième écran va contenir le menu et les zones d'affichage des résultats.

Les types de messages qui peuvent être affichés sont les suivants :

Le premier, concerne le choix de l'opération à effectuer

Taper 1 pour l'addition, 2 pour soustraction, 3 pour la multiplication, 4 pour la division et Q pour quitter.

Le deuxième concerne l'opération elle-même avec son résultat.

Exemple : 1234-543= 991

Le troisième concerne les messages d'erreurs et de fin de programme :

-Si un problème survient lors du traitement d'un fichier, votre programme doit afficher le message d'erreur suivant :

 Erreur dans le traitement du fichier, le programme
 va se terminer

-Si l'affichage se fait correctement, un message demandant à l'utilisateur s'il souhaite faire une autre opération est affiché avant de terminer le programme.

Bon travail !