TD 2 Compilation – Premier contact avec la MV

L3 INFO, Univ Lumière Lyon 2

2022 – 2023

Exercice 1 Rappelez-vous que la commande `man machin` affiche la page d'aide pour la commande `machin`. Appuyez sur la touche `:q` pour sortir de la page d'aide.

1. Connectez-vous à votre machine virtuelle en utilisant la commande `ssh`. Vous êtes l'utilisateur root de la machine. En cas de besoin (ou préférence), vous pouvez utiliser l'image de MV disponible sur moodle [1].
2. Obtenez le répertoire de travail actuel avec la commande `pwd` (*present working directory*).
3. Déplacez-vous vers répertoire temporaire du système de fichiers (commande `cd -` *change directory*), puis créez un dossier nommé `pickabou` à l'aide de la commande `mkdir` (`make directory`).
4. Affichez le contenu du dossier (commande `ls` pour `list`).
5. Affichez la chaîne de caractères "coucou" à l'aide de la commande `echo`.
6. Tapez la commande suivante `echo "machin" > fichier`. Expliquez le résultat (affichez le contenu du dossier et de fichiers présents).
7. Redémarrez votre machine virtuelle.
8. Vérifiez que les dossiers et fichiers que vous avez créé dans le répertoire `/tmp` n'y sont plus.

Exercice 2 Voici les caractères selon l'encodage ASCII (`American Standard Code for Information Interchange`). Les listes sont en hexa et décimale.

1. Pouvez-vous identifier les codages ? Lequel est hexadécimale et lequel est décimale ?
2. Identifiez le code ASCII pour les lettres du mot "pickabou" en hexadécimal et en décimal.
3. Obtenez l'encodage binaire du mot.
4. Vérifiez votre résultat avec la commande `xxd`.
5. Utilisez la commande `man ascii` pour consulter la page d'aide contenant ces tables (tout en bas de la page), et une autre avec l'encodage en binaire. Ces trois tables sont très utiles en pratique.
6. Pouvez-vous dire pourquoi il n'y a pas de colonne 0 et 1 sur la table à gauche ?

```
   2 3 4 5 6 7        30 40 50 60 70 80 90 100 110 120
  -------------      ---------------------------------
0:   0 @ P ` p     0:    (  2  <  F  P  Z  d   n   x
1: ! 1 A Q a q     1:    )  3  =  G  Q  [  e   o   y
2: " 2 B R b r     2:    *  4  >  H  R  \  f   p   z
3: # 3 C S c s     3: !  +  5  ?  I  S  ]  g   q   {
4: $ 4 D T d t     4: "  ,  6  @  J  T  ^  h   r   |
5: % 5 E U e u     5: #  -  7  A  K  U  _  i   s   }
6: & 6 F V f v     6: $  .  8  B  L  V  `  j   t   ~
7: ' 7 G W g w     7: %  /  9  C  M  W  a  k   u   DEL
8: ( 8 H X h x     8: &  0  :  D  N  X  b  l   v
9: ) 9 I Y i y     9: '  1  ;  E  O  Y  c  m   w
A: * : J Z j z
B: + ; K [ k {
C: , < L \ l |
D: - = M ] m }
E: . > N ^ n ~
F: / ? O _ o DEL
```

L'encodage ASCII n'utilise que la moitié d'un octal pour représenter un ensemble relativement réduit de caractères utilisés dans le monde. Beaucoup de place reste disponible, ce qui a conduit à l'existence de plusieurs encodages. Aujourd'hui un seul et unique standard international semble se dégager. Il s'agit d'Unicode, développé par le Consortium Unicode, avec un système très ingénieux basé en un nombre de bits extensible (utf-8, utf-16, ...)

Unicode est un super ensemble d'ASCII, c'est-à-dire qu'on peut récupérer en Unicode les mêmes caractères ASCII en utilisant les mêmes codes.

Pour représenter les caractères en Unicode, on utilise souvent comme notation un code de la forme U+ $\underbrace{1043}_{\textbf{code héxa}}$. Dans l'invite de commandes, nous remplaçons le préfixe U+ par\U.

7. Utilisez la commande `echo -e "\U41"`pour obtenir le caractère ASCII qui correspond à la position 41 (comparez avec l'écriture avec celle utilisée pour l'encodage ASCII).
8. Cherchez sur internet le code hexadécimal de deux ou trois émojis et obtenez leur affichage dans la console.

1. Cette machine est à utiliser avec un logiciel de virtualisation comme VirtualBox

Exercice 3

1. Trouvez le fichier nommé batD dans votre machine virtuelle (utilisez la commande `locate nom_de_fichier`
2. Utilisez la commande `file` pour déterminer le type de fichier.
3. Modifiez le nom du fichier avec la commande `cp source destination` pour rajouter une extension au fichier.
4. Créez une copie cachée du fichier (pour rappel, les fichiers cachés commencent par un point de leur nom, i.e. `.fichier`).

Exercice 4 Utilisez l'éditeur n`nano` pour écrire un fichier contenant le code qui suit

```c
#include <stdio.h>

int main( ) {
  int x = 3 ;
  int y = 2 ;
  int accum = 0;

  accum = x + y ; //sum(x, y);
  printf("%i + %i => %i\n", x, y, accum);

  return(0);
}
```

1. Enregistrez le fichier sous le nom de `acc.c`. Obtenez une première compilation du fichier, en mettant comme nom du fichier binaire de sortie `acc`.
2. Observez la table de liens du programme (commande `ldd`).
3. Exécutez le programme. Il se peut que vous devriez changer les droits du fichier pour le rendre exécutable avec la commande `chmod +x acc`.
4. Obtenez le dump du fichier binaire avec `objdump` et gardez le résultat dans un fichier `acc_dump`. Identifiez la section `main` du code assembler.
5. Obtenez des versions alternatives du fichier binaire en modifiant les options d'optimisation du compilateur (drapeaux `-O[0-2|g]`). Faites bien attention à nommer les différentes versions de manière à pouvoir vous repérer par la suite.
6. Comparez la taille de chaque fichier binaire (commande `ls`) ainsi que les temps d'exécution.
7. Utilisez le drapeau `-i` pour obtenir la version intermédiaire de la compilation qui correspond à la précompilation. Nommez le fichier résultant **acc.i**.
8. Utilisez le drapeau `-s` pour obtenir la version intermédiaire de la compilation qui correspond à l'assamblage. Nommez le fichier résultant **acc.i**.