# TP 2

SYSTEMES D'EXPLOITATION

# INFORMATIONS GÉNÉRALES

- Séance présentation de TP (Mardi 14h-16h)
  - Assistant d'enseignement :
    - Marios Fanourakis: marios.fanourakis@unige.ch
- Séance travail en groupe / individuel pour TP (Mercredi 9h-13h):
  - Si vous avez besoin d'assistance pour un TP pendant cette séance, vous devez en informer les moniteurs et prendre rendez-vous
  - Moniteurs:
    - Rose Defossez: rose.defossez@etu.unige.ch
    - Ethan Arm: ethan.arm@etu.unige.ch
  - Si vous avez besoin d'aide en dehors de ces horaires, contactez l'assistant pour convenir d'un rendez-vous

# DELIVERABLES

- Date de reddition : 17 Octobre à 23:59

- Vous pouvez travailler avec un(e) collègue
- Rapport en pdf
  - Réponses au exercices et manuel de votre programme
- Code source commenté et fonctionnel
- Dossier à rendre :
  <Prenom1>_<Nom1>.<Prenom2>_<Nom2>.TP2.zip(ou tar.gz)

# OBJECTIFS

L'objectif général du TP est de créer un programme qui permet de générer des hashs, également appellés digests (MD5, SHA1, etc.) à partir de chaines de caractères ou de fichiers. Le but des digests est de pouvoir contrôler l'intégrité de données transmises entre deux entités.
Les objectifs de ce TP sont de:

- manipuler des chaines de caractères en C;
- manipuler les paramètres argc / argv de la fonction main;
- se familiariser avec la fonction getopt qui permet de manipuler les paramètres passés au programme;
- effectuer un lien vers des librairies externes (ici la librairie EVP de openssl);
- se familiariser avec les fonctions de hachage qui est un mécanisme de base de la cryptographie.

# CRYPTOGRAPHIE

- authenticité: s'assurer que l'on communique bien avec la personne/machine souhaitée;ˆ
- confidentialité: chiffrer les messages envoyés pour qu'ils ne soient pas lisibles par n'importe qui (i.e. seule une personne possédant une clé de décryptage peut lire le message);ˆ
- intégrité: s'assurer que le message reçu est bien identique au message envoyé et qu'il n'a pas été corrompu lors du transfert.

# HACHAGE

- Il doit être improbable de trouver deux messages ayant le même hash (h(m1) =h(m2));
- une petite modification du message doit donner lieu à un important changement de d=h(m);
- le message ne doit pas pouvoir être reconstruit à partir de la valeur retournée par la fonction de hachage (pour des raisons de sécurité non évoquées ici)

# EXERCICES

**Hachage**

lisez le manuel:

- sha1sum
- Md5sum

- Créez un fichier avec le texte "le manuel disait: Nécessite Windows 7 ou mieux. J'ai donc installé Linux" sans retour à la ligne. Utilisez les commandes de hachage dessus.
- Utilisez le commande "echo" pour le hachage de même texte dessus.
- Pourquoi sont-ils diffèrent? Comment résoudre la problème?

# EXERCICES

**EVP_Digest**

lisez le manuel:

- EVP_Digest (man EVP_DigestInit)
- Implementez l'example dans le manuel
- Compilez le programme en prenant lier les librairies libssl et libcrypto
- Testez le programme

# EXERCICES

**getopt**

lisez le manuel:

- getopt (man 3 getopt)
- Implementez l'example dans le manuel

# PROGRAMME

- Créez un programme pour générer des hash de fichiers et de chaines de caractères.
  - hash –f fichier1 fichier2 …
  - hash ceci est une chaine
  - hash … -t MD5

# PROGRAMME

Le programme devra être implementé en deux modules contenant chacun un .c et un .h ainsi que du fichier .c contenant la fonction main(5 fichiers au total):

- 1er module pour gerer les options (.c et .h)
- 2em module pour calculer les hash (.c et .h)

main.c

```
int main(int argc, char *argv[])
{
 ...
}
```

# MAKEFILE

- Pour compiler votre programme:
  - gcc -o hash main.c digest.c options.c -lssl –lcrypto

- rendez ce processus plus facile pour ceux qui souhaitent compiler votre programme avec un makefile!

Makefile:

```
all: main.c options.c digest.c
        gcc -o digest main.c options.c digest.c -lssl -lcrypto
```