TP 2

Principes des Systèmes d'Exploitation

IUT Montpellier

Objectifs de la séance

- Rappels sur les opérations d'entrées/sorties fichiers
- Implémenter le chargement des 'sprite' dans la libraire du moteur de jeu Ascii

Rappels

- Il existe 5 opérations importantes:
  - open
  - read
  - write
  - ioctrl
  - close
- Vous pouvez consulter les manuels de ces opérations en utilisant la commande 'man'

Objectif

- La commande 'tee' effectue des lectures sur son entrée standard et recopie les octets lus dans la sortie standard *et* dans fichier donné comme argument.
- Nous proposons de réaliser en C un tel programme.

Exercice 1

- Proposez un programme 'mytee' qui réalise la même fonction que 'tee' en utilisant uniquement les appels E/S.

Exercice 2

- Par défaut, la commande tee écrase le contenu du fichier existant. Par contre si la ligne de commande comporte l'option '-a' alors le contenu est ajouté à la fin du fichier. Réalisez cette option dans votre programme.

Objectif

- L'objectif de cet exercice est de réaliser un éditeur de fichier en utilisant uniquement la ligne de commande.

Les spécifications

- Votre programme recevra comme premier paramètre le chemin vers le fichier à éditer et une suite de commandes.
  Pour réaliser cet exercice vous devez utiliser les fonctions d'accès E/S bas niveau (open, read, write, close et lseek)
- Votre programme recevra comme premier paramètre le chemin vers le fichier à éditer et une suite de commandes.
  Voici la liste des commandes que votre programme devra interpréter :
  - s<decalager> : réaliser un décalage du curseur de valeur <decalage> à partir du début du fichier.
  - r<longueur> : lecture de <longueur> octets à partir de la position actuelle du curseur et affichage de ces octets sur la sortie standard
  - w<str> : écriture de la suite de caractères <str> dans le fichier à partir de la position actuelle du curseur.

  Pour réaliser cet exercice vous devez utiliser les fonctions d'accès E/S bas niveau (open, read, write, close et lseek)

Exemple

```
$ touch monfichier
$ stat -l monfichier
-rw-r--r-- 1 x x 0 monfichier
$ ./editor ./monfichier s256 wCouCou
s256: deplacement du curseur reussi
wCouCou: ecriture de 6 bytes
$ stat -l monfichier
-rw-r--r-- 1 x x 262 monfichier
$ ./editor ./monfichier s252 r10
s252: deplacement du curseur reussi
r10: 00 00 00 00 CouCou
$
```

Explications

- On remarque qu'après la création avec la commande touch le fichier a 0 octets comme taille (retour de commande stat)
- L'éditeur effectue un déplacement de 256 octets et ensuite écrit le mot 'CouCou'
- La commande stat cett fois-ci nous donne la taille du fichier qui est de 262 octets (=256 + 6). Il faut remarquer que c'est un fichier qui comporte un 'trou' de 256 octets.
- La dernière ligne demande un déplacement du curseur à l'octet 252
- et une lecture de 10 octets. On affiche bien 6 valeurs null à cause du trou (00 en hexa) et les caractères de la chaine "CouCou"

Exercice 2

- A faire
  - Créer et organiser votre projet
  - Implémenter l'éditeur
  - Tester votre implémentation

Aide

Voici la liste des fonctions C utiles pour la réalisation du programme demandé: open, close, write, read, lseek, strtol, malloc, printf, isprint. Merci de consulter la documentation de ces fonctions pour pouvoir les utiliser.

Présentation du projet

- Pour notre moteur, nous avons fait le choix de représentations en images ASCII.
- Pour cette séance, nous allons intégrer une nouvelle fonctionnalité qui permet de charger une image ASCII à partir d'un fichier sur le disque.
- Dans le jargon du jeu video, une image est une ressource et les ressources sont appelées : 'assets'
- Donc notre but est de réaliser un 'asset loader' qui va lire le contenu d'un fichier externe et le transformer en un format d'image interne.

Introduction

- Le format image ASCII
  - Une image ASCII est simplement un fichier texte avec le format suivant:
  - `<largeur>`
    `<longueur>`
    `<suite de caractères de contenu de l'image; ligne par li`
- Exemple
  - 4
    2
    `{@@}/""\`
  - Représente l'image suivante:
    - `{@@}`
      `/""\`

Implementation

- Préparation
  - Consulter les fichiers suivants:
    - ./src/sprite.h et ./src/sprite.c
    - ./test/testasset/test.c
    - ./asset/image.1.txt
- Développement
  - Compléter l'implémentation de asset.c
- Test
  - Compiler et exécuter le fichier de test
  - Résultat attendu: