Travail pratique de programmation système avancée

Appels système

1 Objectifs

- Utilisation de la FFI
- Utilisation des appels système

2 Énoncé

2.1 Écriture dans un fichier

En utilisant uniquement des appels système FFI depuis la libc directement écrire un programme qui permet d'écrire une chaîne de caractères arbitraire dans un fichier sur le disque qui a également un nom arbitraire.

Commencez par appeler `open()` pour obtenir un descripteur de fichier, `fd`, en créant un nouveau fichier si le fichier n'existe pas ou en l'écrasant s'il existe déjà. Donnez à l'utilisateur les permissions lecture/écriture à l'utilisateur/trice.

Puis utilisez l'appel `write()` pour écrire une chaîne de caractères dans le fichier `fd`. Pensez à bien fermer le fichier après avoir écrit dedans à l'aide de l'appel à `close()`.

Ensuite ouvrez à nouveau le fichier en mode lecture uniquement avec l'appel `open()`. Lisez le contenu du fichier à l'aide de l'appel `read()` et écrivez le résultat dans une mémoire tampon. **Attention** la mémoire tampon pourrait être trop petite alors il faut possiblement faire la lecture **dans une boucle** `loop`. Une fois tout le contenu du fichier lu et copié dans un `Vec<u8>` convertir le résultat en chaîne de chaîne de caractères et l'afficher à l'écran. Pensez à bien fermer à nouveau le fichier à l'aide de l'appel à `close()`.

2.2 Indications

Vous aurez besoin des appels système:

```
#[cfg(target_family = "unix")]
#[link(name = "c")]
extern "C" {
    fn open(pathname: *const i8, flags: i32, mode: i32) -> i32;
    fn write(fd: i32, buf: *const u8, count: usize) -> i32;
    fn read(fd: i32, buffer: *mut u8, cout: usize) -> i32;
```

```
    fn close(fd: i32) -> i32;
}
```

ainsi que des bitmasks

```
const O_RDONLY: i32 = 0;
const O_WRONLY: i32 = 1;
const O_CREAT: i32 = 64;
const O_TRUNC: i32 = 512;
const S_IRUSR: i32 = 0o400;
const S_IWUSR: i32 = 0o200;
```

les `O_*` sont pour les flags, et les `S_*` pour le mode d'ouverture.

Afin d'ouvrir un fichier, vous devez utiliser les types `CString` de `use std::ffi::CString` et sa fonction `as_ptr()` pour convertir le `CString` en `*const i8`.