# Appels système

# Généralités

Les appels système sont des appels aux fonctionnalités du systèmes d'exploitation. Par exemple les appels qui permettent d'accéder au matériel,

les fichiers par exemple open, read, write (entrée/sortie)

la mémoire par exemple malloc, free

les threads par exemple fork, wait

etc.

Sous linux la liste des appels systèmes s'obtient avec la commande

>man syscalls

Des informations générales sont disponibles via

>man 2 intro

# Appels systèmes en C

Le langage C est standardisé et une librairie standard la **libc** (**glibc** pour GNU standard library) est définie. Cette librairie inclut des fonctions répertoriées comme essentiels, dont les appels systèmes.

Les fonction de la libc qui implémentent les appels systèmes sont appelées des **wrapper functions** car elle ne font rien d'autres que d'initialiser les registres du processeurs selon les arguments passés et de faire l'appel système (comment??).

Quand on utilise qemu-arm on exécute la commande

>qemu-arm –L /usr/arm-linux-gnueabi

c'est-à-dire q 'on indique avec l'option –L où se trouve la librairie standard (entres autres). Dans le répertoire on trouve le fichier libc.so, ou un autre nom proche, par exemple libc-2.23.so, que qemu utilise pour trouver les fonctions qu'on appelle dans notre programme (qemu ou alors n'importe quel autre exécutable écrit en C).

# Appels systèmes en C

Notez la terminologie

**arm-linux-gnueabi**

qui veut dire les librairies pour

système ARM

OS = Linux

gnueabi = Gnu embedded application binary interface (comme une API mais on a seulement le code binaire à disposition)

**libc.so**

**so** = shared object c'est le nom des librairies dynamiques sous linux (.dylib sous MAC et .dll sous windows)

# Appels systèmes en C

Pour que le compilateur gcc puisse compiler il doit connaître le prototype des fonctions. Ils se trouvent dans le fichier unistd.h (répertoire /usr/arm-linux-gnueabi/include). L'édition des liens (l'appel à la fonction) se fait dynamiquement pendant l'exécution du programme.

Cherchez la fonction **write** dans votre fichier unistd.h.

# Appels systèmes en ASM

En assembleur comment se font les appels systèmes dépend de l'architecture du système, le processeur.

Sous linux la documentation est disponible via la commande

>man 2 syscall

Pour les architectures arm/EABI les appels systèmes se font avec l'instruction **swi #0** et le numéro de l'appel système se trouve dans r7. Le registre r0 est utilisé comme paramètre de retour.

Où trouver le numéro?

Dans le fichier /usr/arm-linux-eabu/include/asm/unistd.h

L'appel système write à le numéro 4.

# Appels systèmes en ASM

Le prototype de la fonction en C est

**ssize_t write(int** *fd*, **const void** **buf*, **size_t** *count*);

>man 2 write

La fonction écrit *count* bytes depuis le buffer **buf* sur le descripteur de fichier *fd* et retourne le nombre de bytes écrits ou -1 s'il y a une erreur.

Dans unistd.h vous trouvez aussi les descripteurs de fichiers standards, entrée standard = 0, sortie standard = 1, sorte erreur = 2 (à vérifier)

# Exercice

Ecrire en assembleur arm un programme qui affiche à l'écran *hello world* et qui utilise l'appel système **write** pour faire l'affichage.