Miniprojet système : développement d'un noyau de système d'exploitation

Responsable : Christophe Rippert
`Christophe.Rippert@Grenoble-INP.fr`

Mise en place de l'environnement de travail

Introduction

Avant de commencer le miniprojet, il faut mettre en place l'environnement de travail que l'on va utiliser pendant les séances.
Un système d'exploitation s'exécute sur machine nue. Mais pour simplifier le développement et la mise au point des prototypes, nous allons utiliser un émulateur gratuit très populaire appelé QEmu disponible sur la plupart des systèmes. L'intérêt de cet environnement d'exécution est qu'il est portable (vous pouvez l'installer sur vos machines personnelles et travailler en dehors des salles PC) et parfaitement transparent pour votre prototype : il n'y aurait rien à changer dans votre code pour que votre système s'exécute sur une machine nue.
Vous devez tout d'abord récupérer un ensemble de sources de départ que vous devez décompresser dans votre répertoire de travail. Les sources distribuées sont dans le répertoire `src_de_base`, qui contient les sources du noyau et une minibibliothèque C qui vous aidera à développer votre prototype.
La compilation d'un noyau se fait simplement en se plaçant dans le répertoire `src_de_base` et en tapant la commande `make` : si tout se passe bien, le binaire `kernel.bin` est produit, il s'agit d'un exécutable un peu différent de ceux que l'on produit habituellement quand on compile un programme C ou assembleur (vous ne pouvez donc pas l'exécuter directement dans un terminal).

Prise en main de l'environnement de développement

Lorsque vous lancez l'exécution du noyau dans QEmu, un certain nombre d'opérations d'initialisation sont effectuées puis la fonction `kernel_start` localisée dans le fichier `start.c` s'exécute : il s'agit du point d'entrée de votre noyau (comme la fonction `main` dans un programme C classique). Dans les sources fournies, cette fonction commence par un appel à la fonction `fact` qui calcule 5! : il s'agit d'un simple exemple pour vous entraîner à utiliser GDB, vous pourrez supprimer `fact` une fois que vous commencerez le développement de votre noyau.
Pour lancer l'exécution du noyau, deux commandes sont fournies :

- `make run` lance directement l'exécution du noyau sans nous laisser le temps s'y connecter GDB : on ne l'utilisera donc en général pas en phase de mise au point ;
- `make debug` lance QEmu en mode mise au point : c'est cette commande qu'on utilisera le plus souvent.

Note : si pour une raison quelconque vous préférez travailler sans environnement graphique (par exemple si vous vous connectez à distance sur les machines de l'école via une connexion internet très lente), vous pouvez utiliser la commande `make curses` qui est équivalente à `make debug` mais avec un affichage directement dans le terminal.

Mise au point

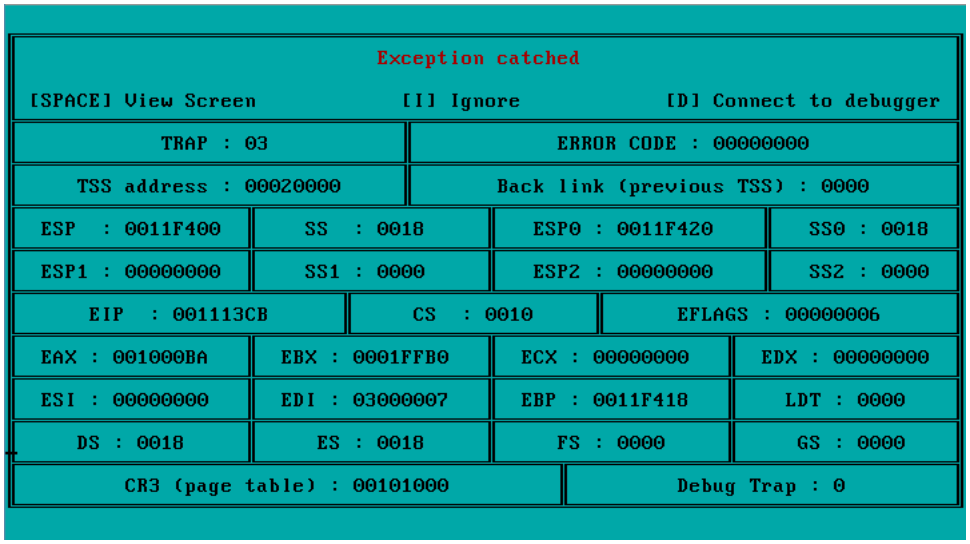
Pour mettre au point le système, il est souvent utile d'utiliser un logiciel comme GDB. QEmu fourni tout ce qu'il faut pour permettre l'exécution pas à pas du système émulé depuis GDB (par contre, il n'y a pas de support pour Valgrind).
Ouvrez un 2e terminal et taper la commande :

```
gdb kernel.bin
```

Cela lance GDB sur le binaire du noyau. Pour connecter le GDB à QEmu, on doit taper dans GDB la commande

```
target remote :1234
```

1234 est ici le numéro du port via lequel QEmu communique avec GDB.
On peut ensuite mettre un point d'arrêt au début du noyau en tapant `b kernel_start` puis lancer l'exécution avec la commande `c` (`continue`).
Le noyau démarre alors et s'arrête au début de `kernel_start`. Vous pouvez utiliser les commandes classiques `s` (`step`), `n` (`next`) et `display` pour afficher le contenu de `x` et exécuter pas à pas la fonction factorielle.
Lors de vos développements, vous ferez vraisemblablement des erreurs d'accès mémoire (déréférencement d'un pointeur nul, accès à une zone interdite, ...). Lorsque cela arrivera, la page d'information ci-dessous s'affichera dans l'écran de QEmu :



Cette page d'information contient notamment :

- le numéro de l'exception (TRAP) et le code d'erreur qui servent à identifier la cause du problème (vous pouvez consulter la liste des exceptions du x86 pour plus d'information) ;
- le contenu des registres du processeur : les plus importants sont le pointeur de pile (ESP), le pointeur d'instruction (EIP) et les registres de segment (principalement CS, DS et SS), leurs valeurs peuvent vous donner une indication sur le problème (e.g. : une valeur de 0 dans un registre-pointeur est rarement une bonne chose !).

Lorsqu'on arrive sur cette page d'information, on peut aussi appuyer sur la touche espace qui permet de basculer entre l'affichage de la page d'information et de l'écran normal du système, pour voir d'éventuelles traces.