L3 – Cours Système – Travaux dirigés
Premiers pas en Nachos

Emmanuel Agullo, Eddy Caron

Vendredi 17 février 2006

1 Threads noyau

Positionnez-vous sous `nachos/code` dans la copie que vous venez d'extraire, et construisez Nachos.

```
cd /home/durand
cvs co nachos
cd nachos/code
make clean      # Pour se remettre dans un état standard (prudent!)
(make depend     # Pour produire les fichiers de dépendances)
make            # Pour lancer la compilation (long...)
```

Normalement, tout doit bien se passer...

Dans un premier temps, nous allons tester les threads systèmes Nachos, c'est-à-dire les threads de niveau *noyau*. Placez-vous dans le répertoire `threads`. Lancez Nachos :

```
./nachos
```

Les options de compilation de ce répertoire font que Nachos lance la fonction `ThreadTest` dans `main.cc` (allez jeter un oeil !). Cette fonction est définie dans `threadtest.cc`. Examinez attentivement sa définition.

Allons maintenant éditer le fichier `threadtest.cc`. Ajoutez le lancement d'un autre thread dans la fonction `ThreadTest()`. Ça marche toujours ?

La sémantique de la méthode `Fork` de l'objet `thread` n'a rien à voir avec celle de la fonction Unix `fork`. Que fait la méthode `fork` dans Nachos ? À quel moment les threads nachos sont-ils créés (mémoire allouée, structures initialisées, ...) ?

Maintenant, commentez la ligne

```
currentThread->Yield();
```

Recompilez (`make`) et examinez ce qui se passe. Qu'en déduisez-vous pour la préemption des threads systèmes par défaut ?

Restaurez cette ligne. On peut lancer Nachos en forçant un certain degré de préemption par l'option `-rs <n>`. De plus, la semence passée en paramètre rend aléatoire (mais *reproductible* !) l'entrelacement des threads.

```
./nachos -rs 0
./nachos -rs 1
./nachos -rs 7
```

Que se passe-t-il ? Couplez cela avec l'option `-d +`. Combien de ticks d'horloge maintenant ?

Vous avez compris ? (C'est assez difficile...) Vérifiez votre intuition en commentant la ligne

```
currentThread->Yield();
```

Vos conclusions ?

2 Programme utilisateur

Déplacez-vous sous le répertoire `test`, et regardez le programme `halt.c`. Faire `make` pour le compiler.

Placez-vous ensuite sous `userprog`. Lancez

```
./nachos -x ../test/halt
```

Bravo : vous venez de lancer votre premier thread qui a vécu 22 cycles machine, dont 10 pour le système et 12 pour le programme...

Essayez de le tracer :

```
./nachos -d + -x ../test/halt

./nachos -rs 0 -d + -x ../test/halt
```

L'option `-d` permet d'activer différents niveaux d'affichage de message de débogage. `-d +` affiche tous les messages tandis que `-d t` affiche les messages à propos des threads, `-d a` les espaces d'adressage, ...

Vous pouvez en plus exécuter le simulateur MIPS pas à pas :

```
./nachos -s -x ../test/halt
```

Vous pouvez toujours observer au plus bas niveau grâce à `gdb` :

```
gdb ./nachos
[...]
run -x ../test/halt
```

Modifiez le programme `halt.c` pour y introduire un peu de calcul, par exemple en faisant quelques opérations sur une variable entière. Tracez pas à pas pour bien voir que cela change quelque chose...

De plus en plus fort ! Détruisez le fichiers `halt.o` et faites `make`. Récupérez la ligne de commande produite, quelque chose comme

```
/home/eagullo/opt/xgcc/decstation-ultrix/bin/gcc -c \
-I../userprog -I../threads -G 0 -c halt.c
```

Changez le `-c` final en `-S`. Cela va produire le code assembleur MIPS de `halt` dans le fichier `halt.s`. Exécutez de nouveau `halt` pas à pas en suivant maintenant les instructions code assembleur !

Bravo ! Vous êtes maintenant prêts pour le grand bain...