Gestion avancée de la pile setjmp/longjmp

Abdelkader Gouaïch

IUT de Montpellier

2014-2015

Gestion avancée de la pile setjmp/longjmp

La pile est une section de l'espace d'adressage utilisée pour les appels de fonctions
On y range :

1. les paramètres de l'appel
2. les variables automatiques de la fonction
3. le contexte de l'appel afin de le restituer au return

- Nous allons maintenant présenter deux fonctions setjmp et longjmp
- Ces deux fonctions vont nous permettre de casser le modèle classique de gestion des appels
- Ces fonctions sont utilisées pour effectuer des sauts (goto) entre des fonctions en cours d'évaluation
- Notre but est de dépiler directement des "fonctions" qui sont dans la pile

Signature des fonctions

```
#include <setjmp.h>
int setjmp(jmp_buf env);
void longjmp(jmp_buf env, int val);
```

- Setjmp : l'appel de cette fonction va établir une cible pour le jump
- La cible est exactement le point où setjump est appelée
- Lors d'un saut avec longjmp c'est l'instruction suivante de setjump qui sera exécutée
- Cela donne l'impression d'un retour de la 'fonction' setjump

Signature des fonctions

```
#include <setjmp.h>
int setjmp(jmp_buf env);
void longjmp(jmp_buf env, int val);
```

- il faut faire la différence entre le premier appel de setjump et un retour d'un saut la fonction
- Astuce : setjump retourne des valeurs différentes !
- Retourne 0 lors du premier appel (mettre en place la cible)
- Retourne un entier (valeur de longjump) si nous venons d'un jump

Signature des fonctions

```
#include <setjmp.h>
int setjmp(jmp_buf env);
void longjmp(jmp_buf env, int val);
```

- La variable env est importante pour comprendre la notion de contexte
- Elle contient toutes les informations nécessaires pour retrouver la cible dans le même état
- Cette variable doit être globale pour être partagée entre les fonctions
- Ou (rare) passée en paramètre.

Signature des fonctions

```
#include <setjmp.h>
int setjmp(jmp_buf env);
void longjmp(jmp_buf env, int val);
```

- Que fait longjmp ?
- Dépiler les frames de toutes les fonctions appelées depuis le setjmp
- Met la valeur val dans le registre de return
- Remet le PC (programme counter) sur l'instruction suivante du setjmp

Restriction sur l'utilisation de setjmp

- setjmp doit être dans l'expression de contrôle principale de : if, switch, while
- Possibilité d'utiliser : $!, ==, !=, <, >$
- Un appel de fonction simple qui ne soit pas utilisé dans une expression

Restriction sur l'utilisation de setjmp

- Ces restrictions sont la conséquence :
- setjmp doit disposer de toutes les informations pour sauvegarder le contexte de la cible
- Etat des registres
- Etat de la pile

Restriction sur l'utilisation de setjmp

Attention : s = setjmp(env) Ne marche pas car ce n'est pas un appel de fonction simple mais un assignement

Erreur d'utilisation de longjmp

Erreur classique à ne pas faire avec longjmp

1. Appeler setjmp(env) pour sauvegarder l'environnement dans une fonction f()
2. Faire un return de f()
3. Faire un longjmp avec env comme contexte ! Mais f() a déjà fait son return ! ! le contexte env n'est plus valide.

Problème de l'optimisation

- Les optimisateurs peuvent réarranger l'emplacement des variables pour les déplacer de la pile vers les registres
- Ceci peut causer des problèmes pour sauvegarder/restaurer le contexte avec setjmp/longjmp
- Afin d'éviter ces problèmes :
- on peut déclarer les variables comme volatile pour éviter tout déplacement vers les registres
- désactiver l'optimisation pour la fonction