Comment faciliter le processus de debugging en tracant la compilation

Bruno Mateu*[1]

[1]IMT Atlantique – IMT Atlantique, IMT Atlantique, IMT Atlantique – France

Résumé

Un processus de compilation moderne comporte plusieurs centaines de passes qui peuvent chacune contenir plusieurs transformations. Ces transformations sont appliquées sur le code de manière sélective, selon les options de configurations données et des propriétés du code en entrée. À un code source et une configuration donnée (contenant éventuellement une graine pour le générateur de nombres aléatoires), le processus de compilation exécuté sera toujours le même. Cependant, une modification - même légère - de la configuration ou du code peut modifier la liste des transformations appliquées sur le code. Les propriétés du code produit dépendent essentiellement des passes appliquées.
Pour cette raison retracer l'histoire d'une instruction telle qu'elle apparaît dans le binaire, c'est-à-dire à quelle(s) ligne(s) de code source elle correspond et quelles sont les transformations subies par celle(s)-ci, est difficile. S'il est possible de connaître la liste des passes appelées lors d'un processus de compilation, on ne peut pas savoir quelles sont les transformations qu'une passe a réalisé sans comparer la représentation interne en entrée et en sortie.

Pour débugger un code, une première étape peut être de désactiver toute optimisation ou obfuscation, afin de travailler sur un code machine le plus proche possible du code source. Cependant, dans certains cas, le bug détecté cessera d'appartaitre une fois ces options désactivées. Dans ce cas, il peut s'agir d'un bug à l'intérieur d'une passe d'optimisation ou d'obfuscation, et il relève donc des développeurs du compilateur de le résoudre. Mais il peut également s'agir d'un bug dans le code source qui ne se manifeste que lorsqu'il interagit avec certainse optimisations et/ou obfuscations.

Dans les deux cas, pour identifier le bug, accéder à l'histoire des instructions pour mieux comprendre comment chaque instruction du code machine a été formée à partir du code source serait un atout, à la fois pour l'utilisateur du compilateur pour l'aider à identifier le bug, et pour le développeur du compilateur, pour l'aider dans le processus de debugging du compilateur lui-même, en particulier lorsque le code produit par le compilateur n'est pas correct.
Dans cette présentation, je souhaite présenter le problème détaillé ci-dessus, présenter les travaux que j'ai effectué sur le compilateur LLVM pour tracer les transformations, puis présenter deux cas d'usages-type de bugs pour lesquels la trace produite peut faciliter le processus de debug.

*Intervenant