Journal : Conversion entre pointeurs de fonctions incompatibles

Posté par **serge_sans_paille** (**page perso**) le 07/11/18 à 10:24.
Tags : **c++**

Posons nous dans le cas suivant (oui ça commence direct)

```
int strange_apply(int (*f)(int) {

    return reinterpret_cast<(int(*)(int, int)>(f)(1, 2);

}
```

Ce code compile avec un warning depuis le dernier gcc, et c'est bien car c'est en fait un *undefined behavior* (cast entre types de fonctions incompatibles dans le cas présent).

Et pourtant on voit pas mal de code comme ça, p.e. dans les sources de LLVM, sous l'hypothèse que si on utilise pas les derniers arguments, ce n'est pas grave.

Je me suis amusé à créer un nouveau `function_cast` qui permet de juste passer les valeurs dont on a besoin pour éviter cet avertissement:

```
#include <utility>
#include <tuple>


template<typename To, typename From>
class FunctionCast;


template<typename ReturnType, typename... ArgumentTypes, typename FromReturnType, typename... FromArgumentTypes>
class FunctionCast<ReturnType(ArgumentTypes...), FromReturnType(*)(FromArgumentTypes...)> {

    using From = FromReturnType (*)(FromArgumentTypes...);

    From from;

    template<typename Args, std::size_t... Is>
    ReturnType apply(Args args, std::index_sequence<Is...>) const {
        return from(std::get<Is>(args)...);
    }

    public:
    FunctionCast(From f) : from(f) {}
    ReturnType operator()(ArgumentTypes... args) const {
        return apply(std::make_tuple(args...), std::make_index_sequence<sizeof...(FromArgumentTypes)>());
    }


};

template<typename To, typename From>
FunctionCast<To, From> function_cast(From f) {
    return {f};
}


int apply(int (*f)(int)) {
    return reinterpret_cast<int(*)(int, int)>(f)(1, 2); // invalid
    return function_cast<int(int, int)>(f)(1, 2);       // valid
}
```

lien godbolt illustrant l'idée. Le concept est assez simple : on empaquète les arguments dans un tuple, puis on applique la fonction en dépaquetant uniquement une partie des arguments (au passage, on doit perdre quelques informations de type, `std::make_tuple` n'est pas très conservateur à ce sujet).

Et voilà, un nouveau cast qui, malheureusement, ne permet pas de créer de nouveaux pointeurs de fonctions mais un foncteur…