Symantec™

Using variable template template without meta programming

Zhihao Yuan

2023/11/9

lld

Symantec

```
class LinkerDriver
{
    template<class ELFT>
    void link(InputArgList& args);
    template<class ELFT>
    void compileBitcodeFiles(bool skipLinkedOutput);
};
```

link passes ELFT down to all these

Symantec

```
template<typename ELFT>
void readSymbolPartitionSection(InputSectionBase* s);

template<class ELFT>
void splitSections();

template<class ELFT>
void createSyntheticSections();

template<class ELFT>
void findKeepUniqueSections(InputArgList& Args);
```

link's body is huge

Symantec

```
template<class ELFT>
void LinkerDriver::link(InputArgList& args)
{
    compileBitcodeFiles<ELFT>(skipLinkedOutput);
    ...;
    std::erase_if(inputSections,
        [](InputSectionBase* s)
        {
            if (s->type == SHT_LLVM_SYMPART)
            {
              readSymbolPartitionSection<ELFT>(s);
              return true;
            }
```

```
splitSections<ELFT>();
markLive<ELFT>();
createSyntheticSections<ELFT>();

if (config->icf != ICFLevel::None)
{
    findKeepUniqueSections<ELFT>(args);
```

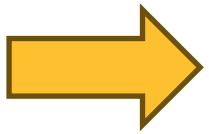
Input to link isn’t compile-time!

Symantec

```
switch (config->ekind)
{
case ELF32LEKind:
    link<ELF32LE>(args);
    break;
case ELF32BEKind:
    link<ELF32BE>(args);
    break;
case ELF64LEKind:
    link<ELF64LE>(args);
    break;
case ELF64BEKind:
    link<ELF64BE>(args);
    break;
default:
    std::unreachable("unknown Config->EKind");
}
```

Goal

```
switch (config->ekind)
{
case ELF32LEKind:
    link<ELF32LE>(args);
    break;
case ELF32BEKind:
    link<ELF32BE>(args);
    break;
case ELF64LEKind:
    link<ELF64LE>(args);
    break;
case ELF64BEKind:
    link<ELF64BE>(args);
    break;
default:
    std::unreachable("unknown Config->EKind");
}
```

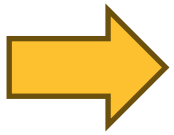


```
link(args);
```

Will this work?

Symantec

```
compileBitcodeFiles<ELFT>(skipLinked);
readSymbolPartitionSection<ELFT>(s);
splitSections<ELFT>();
markLive<ELFT>();
createSyntheticSections<ELFT>();
findKeepUniqueSections<ELFT>(args);
```



```
invokeELFT<compileBitcodeFiles>(skipLinked);
invokeELFT<readSymbolPartitionSection>(s);
invokeELFT<splitSections>();
invokeELFT<markLive>();
invokeELFT<createSyntheticSections>();
invokeELFT<findKeepUniqueSections>(args);
```

Reference to function, templated

Can be done today

Symantec

```
template<class ELFT>
void findKeepUniqueSections(InputArgList& Args);

template<class ELFT>
inline constexpr decltype((
    findKeepUniqueSections<ELFT>)) findKeepUniqueSections_ =
    findKeepUniqueSections<ELFT>;
```

Applicable to pointer-to-members

Symantec

```
template<class ELFT>
inline constexpr decltype((
    &LinkerDriver::compileBitcodeFiles<ELFT>)) compileBitcodeFiles_ =
    &LinkerDriver::compileBitcodeFiles<ELFT>;
```

Conclusion from that experiment

Symantec

- C++ cannot pass X down to the next abstraction layer such that X<A...>(B...) is a call

In the end...

Symantec

```
#define invokeELFT(f, ...)                                  \
    switch (config->ekind) {                                \
    case ELF32LEKind:                                       \
        f<ELF32LE>(__VA_ARGS__);                            \
        break;                                              \
    case ELF32BEKind:                                       \
        f<ELF32BE>(__VA_ARGS__);                            \
        break;                                              \
    case ELF64LEKind:                                       \
        f<ELF64LE>(__VA_ARGS__);                            \
        break;                                              \
    case ELF64BEKind:                                       \
        f<ELF64BE>(__VA_ARGS__);                            \
        break;                                              \
    default:                                                \
        llvm_unreachable("unknown config->ekind");          \
    }
```

But this is what we want

Symantec

```
template<template<class> auto f> void invokeELFT(f, auto& args...) {
    switch (config->ekind) {
    case ELF32LEKind:
        f<ELF32LE>(args...);
        break;
    case ELF32BEKind:
        f<ELF32BE>(args...);
        break;
    case ELF64LEKind:
        f<ELF64LE>(args...);
        break;
    case ELF64BEKind:
        f<ELF64BE>(args...);
        break;
    default:
        llvm_unreachable("unknown config->ekind");
    }
}
```

Thank you