Конструирование компиляторов

affeeal

14 февраля 2024 г.

Содержание

1 14 февраля 2024 г.

1.1 Дополнение к прошлой лекции

Преимущества интерпретатора перед компилятором:

- Выполнение "на ходу".
- Переносимость.

Преимущества комплятора перед интерпретатором:

- Быстродействие.
- Проверка корректности программы на этапе компиляции.

Примеры. Java — и компилируемый, и интерпретируемый ЯП. Так обеспечивается и переносимость, и быстродействие. Порождение Python'ом промежуточного byte-code'а.

JIT-компиляция используется JVM, ADO.NET, Julia.

CISC, RISC — обогащённые и сокращённые системы команд соответственно. ARM реализует RISC в мобильных телефонах. RISC более передовой. См. микрокодовые команды *mu*-code).

1.2 Фазы компиляции

См. SSA-форма (static symbol assignment).

Сопрограммы в отличие от подпрограмм могут приостанавливать свою работу и возвращаться с того же места.

`Примеры`. Go-рутины могут работать и как сопрограммы, и как потоки. Операторы `yield` в Python и `yieldreturn` в C# реализуют сопрограммы.

Компоновка — построение готовой программы из раздельно компилирующихся частей. Выполняется компоновщиом (linker, link editor, "линковщик").

Компоновка бывает

- Статической — до запуска программы.

  Пример. Статическая компоновка для C/C++. На входе исходники на C/C++. Каждый из исходников обрабатывается препроцессором. Результат обрабатывается компилятором. Между препроцессором и компилятором промежуточных файлов, как правило, не создаётся. В Unix-подобных системах компилятор порождает ассемблерный листинг. Из ассемблера порождается объектный файл `.o`. С остальными файлами происходит то же самое. Далее включается компоновщик, получающий на входе объектные фалйы (а также библиотеки, традиционно `.a` в Unix-подобных системах). На выходе компоновщик порождает исполняемый файл с именем по умолчанию `a.out`.

  Библиотека `libm.a`, например, подключается опцией `-lm`.

На Windows — несколько исходников, препроцессор, компилятор. Компилятор сразу порождает объектный файл `.obj`. Библиотеки `.lib` (их тоже может быть несколько). Всё это падает на компоновщик, порождающий исполняемый файл (`.exe`). Т.е. нет фазы ассемблирования.

- Динамической — во время выполнения программы.

  На входе исполняемый файл Unix или Windows, динамические библиотеки `.so`, (`.dll`). Загрузчик ОС создаёт пустое адресное пространство, загружает исполняемые файлы, ищет библиотеки, разрешает перекрёстные ссылки... В результате — процесс в ОЗУ.

Напомнить: библиотека поддержки во время исполнения (real-time support library).