Автоматизация сборки проекта, утилита make. Сценарий сборки проекта. Простойсценарий сборки.
Использование переменных и комментариев. Сборка программы с разными параметрами компиляции.
План ответа:
1 автоматизация сборки проекта: основные задачи, «исходные» данные;
2 разновидности утилиты make;
3 сценарий сборки проекта: название файла, его структура;
4 автоматические переменные;
5 шаблонные правила.

1 автоматизация сборки проекта: основные задачи, «исходные» данные;
Позволяет распределить работу над проектом между несколькими программистами.
Код программы более удобочитаем.
Сокращает время повторной компиляции.
1.1. Задача автоматизации сборки проекта — избавить программиста от необходимости каждый раз печатать объёмные вызовы компилятору и компоновщику в весьма больших проектах.
1.2. Данными для автоматической сборки являются файлы заголовков, реализации и библиотеки (вход), исполняемые файлы и библиотеки (выход).
1.3. Для автоматической сборки проекта применяют несколько способов:
BAT-файлы
Специализированные программные средства сборки (т.н. make)

2 разновидности утилиты make;
2.1. make — утилита, автоматизирующая процесс преобразования файлов из одной формы в другую.
2.2. Известны следующие разновидности средств автоматической сборки проекта (т.н. make):
GNU Make (мы им пользовались)
BSD Make
Microsoft Make (nmake)

3 сценарий сборки проекта: название файла, его структура;
Необходимо создать так называемый сценарий сборки проекта (make-файл).
Этот файл описывает отношения между файлами программы; содержит команды для обновления каждого файла.
Утилита make использует информацию из make-файла и время последнего изменения каждого файла для того, чтобы решить, какие файлы нужно обновить.
Утилита make предполагает, что по умолчанию сценарий сборки называется makefile или Makefile.
цель: зависимость_1 ... зависимость_n
команда_1
команда_2

4 автоматические переменные;
Автоматические переменные - это переменные со специальными именами, которые «автоматически» принимают определенные значения перед выполнением описанных в правиле команд.
Переменная "$^" означает "список зависимостей".
Переменная "$@" означает "имя цели".
Переменная "$<" является просто первой зависимостью.

```
test_greeting.exe : $(OBJS) test.o
        $(CC) $^ -o $@
```

5 шаблонные правила.
%.[расширения целей]: %.[расширения аргументов] — для всех файлов с данной маской поимённо
*.[расширение] — все файлы с данным расширением.

```
%.o : %.c *.h
        $(CC) $(CFLAGS) -c $<
```