**Вопрос 1.**

**Понятие и структура системы программирования**

Всякий компилятор является составной частью системного программного обеспечения. Основное назначение компиляторов служить для разработки новых прикладных и системных программ с помощью языков высокого уровня.  
Любая программа, как системная, так и прикладная, проходит этапы жизненного цикла, начиная от проектирования и вплоть до внедрения и сопровождения. Акомпиляторы это средства, служащие для создания программного обеспечения на этапах кодирования, тестирования и отладки.  
Однако сам по себе компилятор не решает полностью всех задач, связанных с разработкой новой программы. Средств только лишь компилятора недостаточно для того, чтобы обеспечить прохождение программой указанных этапов жизненного цикла. Поэтому компиляторы это программное обеспечение, которое функционирует в тесном взаимодействии с другими техническими средствами, применяемыми на данных этапах.

Основные технические средства, используемые в комплексе с компиляторами, включают в себя следующие программные модули:  
- текстовые редакторы, служащие для создания текстов исходных программ;  
- компоновщики, позволяющие объединять несколько объектных модулей, порождаемых компилятором, в единое целое;  
- библиотеки прикладных программ, содержащие в себе наиболее часто используемые функции и подпрограммы в виде готовых объектных модулей;  
- загрузчики, обеспечивающие подготовку готовой программы к выполнению;  
- отладчики, выполняющие программу в заданном режиме с целью поиска, обнаружения и локализации ошибок.  
Первоначально компиляторы представляли собой обособленные программные модули, решающие исключительно задачу перевода исходного текста программы на входном языке в язык машинных кодов. Компиляторы разрабатывались вне связи с другими техническими средствами, с которыми им приходилось взаимодействовать. Взадачу разработчика программы входило обеспечить взаимосвязь всех используемых технических средств:  
- подать входные данные в виде текста исходной программы на вход компилятора;  
- получить от компилятора результаты его работы в виде набора объектных файлов;  
- подать весь набор полученных объектных файлов вместе с необходимыми библиотеками подпрограмм на вход компоновщику;  
- получить от компоновщика единый файл программы (исполняемый файл) и подготовить его к выполнению с помощью загрузчика;  
- поставить программу на выполнение, при необходимости использовать отладчик для проверки правильности выполнения программы.

Все эти действия выполнялись с помощью последовательности команд, инициировавших запуск соответствующих программных модулей с передачей им всех необходимых параметров. Параметры передавались каждому модулю в командной строке и представляли собой набор имен файлов и настроек, реализованных в виде специальных ключей. Пользователи могли выполнять эти команды последовательно вручную, а с развитием средств командных процессоров ОС они стали объединять их в командные файлы.  
Со временем разработчики компиляторов постарались облегчить труд пользователей, предоставив им все необходимое множество программных модулей в составе одной поставки компилятора. Теперь компиляторы поставлялись уже вкупе со всеми необходимыми сопровождающими техническими средствами. Кроме того, были унифицированы форматы объектных файлов и файлов библиотек подпрограмм. Теперь разработчики, имея компилятор от одного производителя, могли в принципе пользоваться библиотеками и объектными файлами, полученными от другого производителя компиляторов.

Для написания командных файлов компиляции был предложен специальный командный язык язык Makefile. Он позволял в достаточно гибкой и удобной форме описать весь процесс создания программы от порождения исходных текстов до подготовки ее к выполнению. Это было удобное, но достаточно сложное техническое средство, требующее от разработчика высокой степени подготовки и профессиональных знаний, поскольку сам командный язык Makefile был по сложности сравним с простым языком программирования. Язык Makefile стал стандартным средством, единым для компиляторов всех разработчиков.  
Такая структура средств разработки существовала достаточно долгое время, а в некоторых случаях она используется и по сей день (особенно при создании системных программ). Ее широкое распространение было связано с тем, что сама по себе вся эта структура средств разработки была очень удобной при пакетном выполнении программ на компьютере, что способствовало ее повсеместному применению в эпоху mainframe.

Следующим шагом в развитии средств разработки стало появление так называемой интегрированной среды разработки. Интегрированная среда объединила в себе возможности текстовых редакторов исходных текстов программ и командный язык компиляции. Пользователь (разработчик исходной программы) теперь не должен был выполнять всю последовательность действий от порождения исходного кода до его выполнения, от него также не требовалось описывать этот процесс с помощью системы команд в Makefile. Теперь ему было достаточно только указать в удобной интерфейсной форме состав необходимых для создания программы исходных модулей и библиотек. Ключи, необходимые компилятору и другим техническим средствам, также задавались в виде интерфейсных форм настройки.  
После этого интегрированная среда разработки сама автоматически готовила всю необходимую последовательность команд Makefile, выполняла их, получала результат и сообщала о возникших ошибках при их наличии. Причем сам текст исходных модулей пользователь мог изменить здесь же, не прерывая работу с интегрированной средой, чтобы потом при необходимости просто повторить весь процесс компиляции.

Создание интегрированных сред разработки стало возможным благодаря бурному развитию персональных компьютеров и появлению развитых средств интерфейса пользователя (сначала текстовых, а потом и графических). Их появление на рынке определило дальнейшие развитие такого рода технических средств. Пожалуй, первой удачной средой такого рода можно признать интегрированную среду программирования Turbo Pascal на основе языка Pascal производства фирмы Borland. Ее широкая популярность определила тот факт, что со временем все разработчики компиляторов обратились к созданию интегрированных средств разработки для своих продуктов.  
Развитие интегрированных сред несколько снизило требования к профессиональным навыкам разработчиков исходных программ. Теперь в простейшем случае от разработчика требовалось только знание исходного языка (его синтаксиса и семантики). При создании прикладной программы ее разработчик мог в простейшем случае даже не разбираться в архитектуре целевой вычислительной системы.

Дальнейшее развитие средств разработки также тесно связано с повсеместным распространением развитых средств графического интерфейса пользователя. Такой интерфейс стал неотъемлемой составной частью многих современных ОС и так называемых графических оболочек. Со временем он стал стандартом де-факто практически во всех современных прикладных программах.  
Это не могло не сказаться на требованиях, предъявляемых к средствам разработки программного обеспечения. Вих состав были сначала включены соответствующие библиотеки, обеспечивающие поддержку развитого графического интерфейса пользователя и взаимодействие с функциями API (application program interface, прикладной программный интерфейс операционных систем). Азатем для работы с ними потребовались дополнительные средства, обеспечивающие разработку внешнего вида интерфейсных модулей. Такая работа была уже более характерна для дизайнера, чем для программиста.

Для описания графических элементов программ потребовались соответствующие языки. На их основе сложилось понятие ресурсов(1) (resources) прикладных программ.

Ресурсами прикладной программы будем называть множество данных, обеспечивающих внешний вид интерфейса пользователя этой программы, и не связанных напрямую с логикой выполнения программы. Характерными примерами ресурсов являются: тексты сообщений, выдаваемых программой; цветовая гамма элементов интерфейса; надписи на таких элементах, как кнопки и заголовки окон ит.п.  
Для формирования структуры ресурсов в свою очередь потребовались редакторы ресурсов, а затем и компиляторы ресурсов, обрабатывающие результат их работы(2). Ресурсы, полученные с выхода компиляторов ресурсов, стали обрабатываться компоновщиками и загрузчиками.

Весь этот комплекс программно-технических средств в настоящие время составляет новое понятие, которое здесь названо системой программирования.