4. Подключаемые файлы

Подключаемый файл это файл, содержащий определения функций и переменных, а также макроопределения вместе с некоторыми исходными файлами. Для использования в программе подключаемых файлов применяется директива препроцессора '#include'.

4.1 Использование подключаемых файлов.

Подключаемые файлы используются для двух целей:

- Системные подключаемые файлы используются для определения интерфейсов к составляющим операционной системы. Они подключаются для предоставления объявлений и определений, требуемых для работы с системными вызовами и библиотеками.
- Подключаемые файлы пользователя содержат определения для интерфейсов между исходными файлами программы.

Включение подключаемого файла в программу дает такой же результат, как при копировании этого файла в каждый исходный файл этой программы. Подобное копирование занимает много времени и побуждает возникновение ошибок. При использовании подключаемых файлов все объявления и определения переменных и функций находятся в одном файле и при необходимости могут быть изменены.

Обычно подключаемые файлы заканчиваются на '.h' и следует избегать использования других стандартов.

4.2 Директива '#include'.

Как файлы пользователя, так и системные файлы включаются в программу с использованием директивы препроцессора '#include'. Она имеет три модификации:

```
'#include <FILE>'
```

Эта модификация используется для подключения системных файлов. При ее выполнении производится поиск файла с именем FILE в списке указанных заранее каталогов, а затем в стандартном списке системных каталогов. С помощью опции '-I' указываются каталоги для поиска подключаемых файлов. Опция '-nostdinc' запрещает поиск в стандартных системных каталогах и производит поиск только в указанных каталогах.

Синтаксис такой модификации директивы '#include' довольно специфичен, потому как комментарии внутри '<...>' не распознаются. Поэтому в строке '#include *y>' последовательность символов '*' не начинает комментарий, а указанная директива включает в программу файл с именем 'x/*y'.

Аргумент FILE не может содержать символа '>', хотя он может содержать символ '<'.

```
'#include "FILE"'
```

Эта модификация применяется для подключаемых файлов для программ пользователя. Сначала файл FILE просматривается в текущем каталоге, а затем в каталогах для системных подключаемых файлов. Текущим каталогом является каталог текущего обрабатываемого файла. Он просматривается в первую очередь, так как предполагается, что в нем находятся файлы, имеющие отношение к текущему обрабатываемому файлу. (Если указана опция '-I-', то текущий каталог не просматривается.)

Аргумент FILE не может содержать символов '"'. Символы backslash интерпретируются как отдельные символы, а не начало escape последовательности. Таким образом, директива '#include "x\n\\y"' указывает имя файла, содержащего три символа backslash.

```
'#include ANYTHING ELSE'
```

Эта модификация называется "вычисляемой директивой #include". Любая директива '#include', не соответствующая ни одной из модификаций, рассмотреных выше, является вычисляемой директивой. Строка ANYTHING ELSE проверяется на наличие соответствующего макроса, значение которого затем заменяет его название. Полученная в результате строка должна уже в точности соответствовать одной из рассмотренных выше модификаций (то есть имя подключаемого файла должно быть заключено в кавычки или угловые скобки).

Эта возможность позволяет определять макросы, что дает возможность изменять имена подключаемых файлов. Эта возможность, например, используется при переносе программ с одной операционной системы на другие, где требуются разные подключаемые файлы.

4.3 Как работает директива '#include'

Директива '#include' указывает C препроцессору обработать указанный файл перед обработкой оставшейся части текущего файла. Информация, выдаваемая препроцессором, содержит уже полученные данные, за которыми следуют данные, получаемые при обработке подключаемого файла, а за которыми, в свою очередь, следуют данные, получаемые при обработке текста, следующего после директивы '#include'. Например, дан следующий подключаемый файл 'header.h':

```
char *test ();
```

и основная программа с именем 'program.c', использующая этот файл.

```
int x;
#include "header.h"

main ()
{
  printf (test ());
}
```

Данные, полученные при обработке программы 'program.c' будут выглядеть следующим образом:

```
int x;
char *test ();

main ()
{
```

```
  printf (test ());
}
```

Для подключаемых файлов нет ограничений на объявления и макроопределения. Любой фрагмент C программы может быть включен в другой файл. Подключаемый файл может даже содержать начало выражения, заканчивающееся в исходном файле или окончание выражения, начало которого находится в исходном файле. Хотя комметарии и строковые константы не могут начинаться подключаемом файле и продолжаться в исходном файле. Не завершенный комментарий, стороковая или символьная константа в подключаемом файле приводят к возникновению ошибки в конце файла.

Подключаемый файл может содержать начало или окончание сиснтаксической конструкции, такой как определение функции.

Срока, следующая за директивой '#include' всегда является пустой и добавляется C препроцессором даже если подключаемый файл не содержит завершающий символ перевода строки.

4.4 Однократно подключаемые файлы

Часто случается, что подключаемый файл включает в себя другой файл. Это может привести к тому, что отдельный файл будет подключаться неоднократно, что может привести к возникновению ошибок, если файл определяет типы структур или определения типов. Поэтому следует избегать многократного подключения файлов.

Обычно это достигается путем заключения в условие всего содержимого этого файла, как показано ниже:

```
#ifndef FILE_FOO_SEEN
#define FILE_FOO_SEEN

Сам файл

#endif /* FILE_FOO_SEEN */
```

Макрос 'FILE_FOO_SEEN' указывает на то, что файл уже однажды вкючался. В подключаемых файлах пользователя макрос не должен начинаться с символа '_'. В системных подключаемых файлах его имя не должно начинаться с символа '__' во избежание возникновения конфликтов с программами пользователя. Каким бы ни был файл, имя макроса должно содержать имя файла и некоторый дополнительный текст во избежание вознкновения конфликтов с другими подключаемыми файлами.

Препроцессор GNU C построен таким образом, что обработке подключаемого файла он проверяет наличие определенных конструкций и наиболее рационально их обрабатывает. Препроцессор специально отмечает полное вложение файла в условие '#ifndef'. Если в подключаемом файле содержится директива '#include', указывающая на обрабатываемый файл, или макрос в директиве '#ifndef' уже определен, то обрабатываемый файл полностью игнорируется.

Существует также специальная директива, указывающая препроцессору, что файл должен быть включен не более одного раза. Эта директива называется '#pragma once'. Она использовалась в дополнение к директиве '#ifndef' и в настоящее время она устарела и не должна прменяться.

В объектно ориентированном языке C существует модификация директивы '#include', называемая '#import', которая используется для включения файла не более одного раза. При использовании директивы '#import' вместо '#include' не требуется наличия условных оборотов для предотвращения многократной обработки файла.

4.5 Подключаемые файлы и наследование

"Наследование" это то, что происходит, когда какой либо объект или файл образует некоторую часть своего содержимого путем виртуального копирования из другого объекта или файла. В случае подключаемых C файлов наследование означает, что один файл включает другой файл, а затем заменяет или добавляет что-либо.

Если наследуемый подключаемый файл и основной подключаемый файл имеют различные имена, то такое наследование называется прямым. При этом используется конструкция '#include "BASE"' в наследуемом файле.

Иногда необходимо чтобы у наследуемого и основного подключаемого файла были одинаковые имена.

Например, предположим, что прикладная программа использует системный подключаемый файл 'sys/signal.h', но версия файла '/usr/include/sys/signal.h' на данной системе выполняет того, что требуется в прикладной программе. Будет удобнее определить локальную версию, возможно с именем '/usr/local/include/sys/signal.h' для замены или добавления к версии, поставляемой с системой.

Это можно выполнить с применением опции '-I.' при компиляции, а также созданием файла 'sys/signal.h' который выполняет требуемые программе функции. Но сделать так, чтобы этот файл включал стандартный файл 'sys/signal.h' не так просто. При включении строки '#include <sys/signal.h>' в этот файл произойдет подключение новой версии файла, а не стандартной системной версии. Это приведет к рекурсии и ошибке при компиляции.

При использовании директивы `#include </usr/include/sys/signal.h>' нужный файл будет найден, но этот способ является не эфективным, так как содержит полный путь к системному файлу. Это может отразиться на содержании системы, так как это означает, что любые изменения местоположения системных файлов потребуют дополнительных изменений где-либо еще.

Более эффективным решением этой проблемы является применение директивы '#include_next', которая используется для подключения следующего файла с таким же именем. Эта директива функционирует также как и директива '#include' за исключением поиска требуемого файла. Она начинает поиск списка каталогов подключаемых файлов после каталога, где был найден текущий файл.

Предположим была указана опция '-I /usr/local/include', а список каталогов для поиска включает '/usr/include'. Также предположим, что оба каталога содержат файл с именем 'sys/signal.h'. Директива '#include <sys/signal.h>' найдет нужный файл под каталогом '/usr/local/include'. Если этот файл содержит строку '#include_next <sys/signal.h>', то поиск будет возобновлен после предыдущего каталога и будет найден файл в каталоге '/usr/include'.

Вперед Назад Содержание