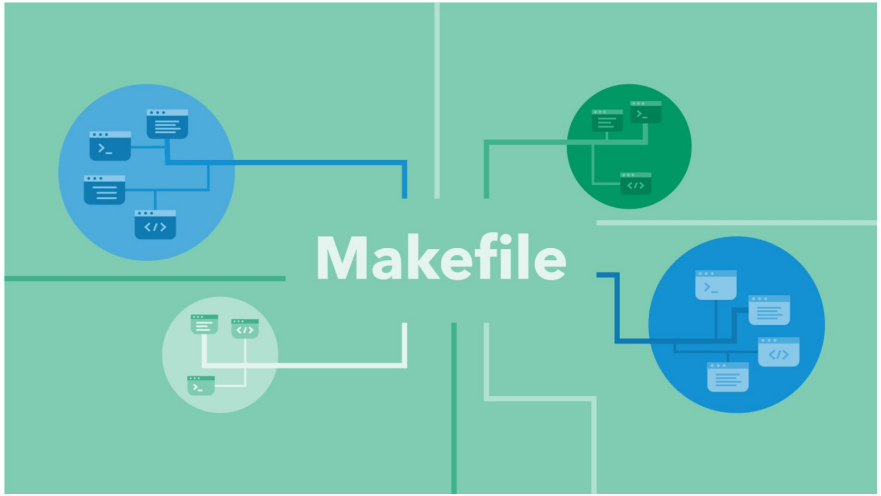
Make файлы

Стазаев Даниил
6.2 группа 3 курс

Make файлы

Make-файлы используются для того, чтобы помочь решить, какие части большой программы необходимо перекомпилировать. В подавляющем большинстве случаев компилируются файлы C++. Другие языки обычно имеют свои собственные инструменты, которые служат той же цели, что и Make. Make также можно использовать и вне компиляции, когда нужно выполнить ряд инструкций в зависимости от того, какие файлы были изменены.



Синтаксис

- Целевыми объектами являются имена файлов, разделенные пробелами. Как правило, для каждого правила существует только одно.
- Команды представляют собой последовательность шагов, обычно используемых для создания целей. Они должны начинаться с символа табуляции, а не с пробелов.
- Обязательными условиями также являются имена файлов, разделенные пробелами. Эти файлы должны существовать до запуска команд для целевого объекта. Они также называются зависимостями.

Простой Makefile

Код программы:

```cpp
#include <iostream>

using namespace std;

int main(){
    cout << "Hello World!"<< endl;
    return 0;
}
```

Makefile:

```
all:
	g++ helloWorld.cpp -o hello
```

Запуск сборки будет выглядеть следующим образом:

```
deeeesp@deeeesp:~/programs/TC/makefile/helloWorld$
make
g++ helloWorld.cpp -o hello
deeeesp@deeeesp:~/programs/TC/makefile/helloWorld$
./hello
Hello World!
```

Использование зависимостей

Make выбирает цель counter

Для counter требуется main.o, factorial.o, hello.o, поэтому выполните поиск каждого элемента

Для main.o требуется main.cpp, у которого нет зависимостей, поэтому выполняется компиляция

Для factorial.o требуется factorial.cpp, у которого нет зависимостей, поэтому выполняется компиляция

Для hello.o требуется hello.cpp, у которого нет зависимостей, поэтому выполняется компиляция

Выполняется компиляция counter, потому что все зависимости завершены

Вот и все: counter - это скомпилированная программа на c++

```
all: counter

counter: main.o factorial.o hello.o
	g++ main.o factorial.o hello.o -o counter-2

main.o: main.cpp
	g++ -c main.cpp

factorial.o: factorial.cpp
	g++ -c factorial.cpp

hello.o: hello.cpp
	g++ -c hello.cpp
```

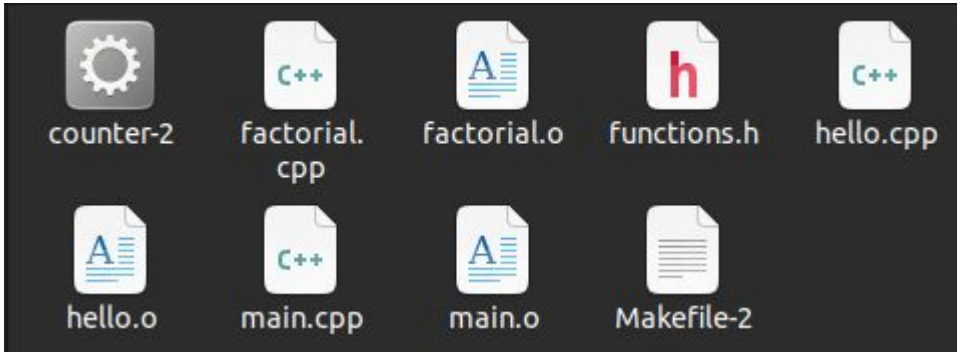
Запуск сборки

В команде: make -f Makefile-2 флаг -f используется, чтобы указать на makefile, который надо использовать при сборке.

При сборке сохраняются файлы:

- factorial.o
- hello.o
- main.o

```
deeeesp@deeeesp:~/programs/tets/mf/counter$ make -f Makefile-2
g++ -c factorial.cpp
g++ -c hello.cpp
g++ main.o factorial.o hello.o -o counter-2
deeeesp@deeeesp:~/programs/tets/mf/counter$ ./counter-2
Welcome to the factorial counter program!
The factorial of 5 is 120
```



Использование цели - clean

Clean традиционно используется для быстрой очистки всех результатов сборки проекта.

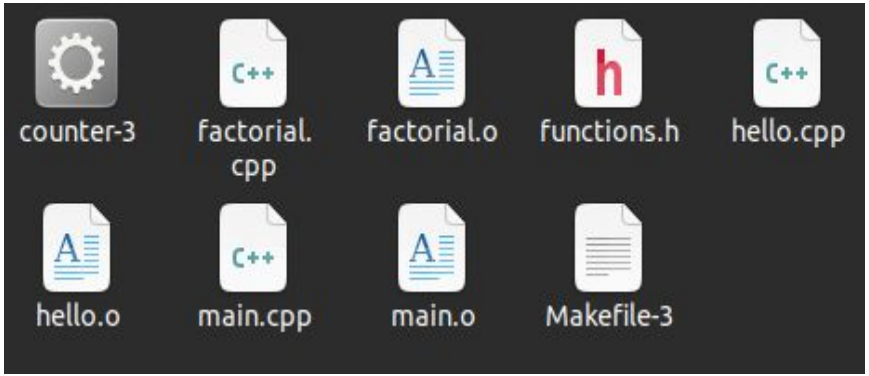
Очистка запускается следующей командой:

make -f Makefile-3 clean

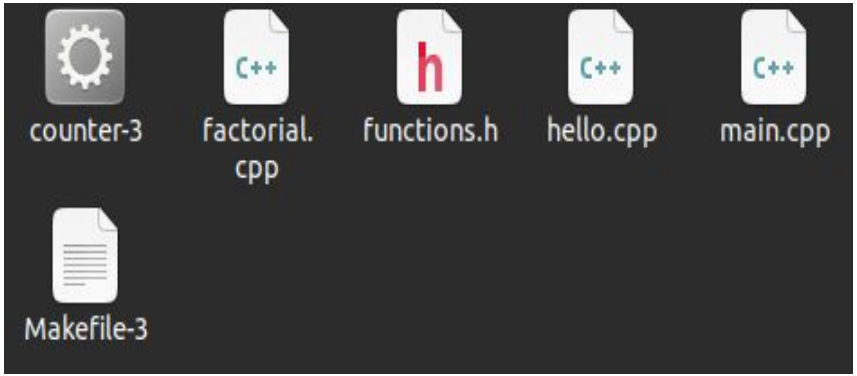
```
all: counter

counter: main.o factorial.o hello.o
	g++ main.o factorial.o hello.o -o counter-3

main.o: main.cpp
	g++ -c main.cpp

factorial.o: factorial.cpp
	g++ -c factorial.cpp

hello.o: hello.cpp
	g++ -c hello.cpp

clean:
	rm -rf *.o
```

Использование clean

```
deeeesp@deeeesp:~/programs/tets/mf/counter$ make -f Makefile-3

g++ -c main.cpp
g++ -c factorial.cpp
g++ -c hello.cpp
g++ main.o factorial.o hello.o -o counter-3
deeeesp@deeeesp:~/programs/tets/mf/counter$ ./counter-3
Welcome to the factorial counter program!
The factorial of 5 is 120
```




```
deeeesp@deeeesp:~/programs/tets/mf/counter$ make -f Makefile-3

g++ -c main.cpp
g++ -c factorial.cpp
g++ -c hello.cpp
g++ main.o factorial.o hello.o -o counter-3
deeeesp@deeeesp:~/programs/tets/mf/counter$ ./counter-3
Welcome to the factorial counter program!
The factorial of 5 is 120
deeeesp@deeeesp:~/programs/tets/mf/counter$ make -f Makefile-3
 clean
rm -rf *.o counter-4
```



Использование переменных и комментариев

```
1  # Это комментарий, который говорит, что переменная CC указывает
   компилятор, используемый для сборки
2  CC=g++
3  #Это еще один комментарий. Он поясняет, что в переменной CFLAGS
   лежат флаги, которые передаются компилятору
4  CFLAGS=-c -Wall
5
6  all: counter
7
8  counter: main.o factorial.o hello.o
9          $(CC) main.o factorial.o hello.o -o counter-4
10
11 main.o: main.cpp
12         $(CC) $(CFLAGS) main.cpp
13
14 factorial.o: factorial.cpp
15         $(CC) $(CFLAGS) factorial.cpp
16
17 hello.o: hello.cpp
18         $(CC) $(CFLAGS) hello.cpp
19
20 clean:
21         rm -rf *.o
```

Использование автоматических переменных

Самые распространенные автоматические переменные:

- $@ Имя цели обрабатываемого правила
- $< Имя первой зависимости обрабатываемого правила
- $^ Список всех зависимостей обрабатываемого правила

```
CC=g++
CFLAGS=-c -Wall
LDFLAGS=
SOURCES=main.cpp hello.cpp factorial.cpp
OBJECTS=$(SOURCES:.cpp=.o)
EXECUTABLE=counter

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
	$(CC) $(LDFLAGS) $(OBJECTS) -o $@

.cpp.o:
	$(CC) $(CFLAGS) $< -o $@
```