Kompilacja programu

Wywołanie: “g++ <option flags> <file list>”

Gdy podajemy g++ zbiór o rozszerzeniu .cpp , to kompilator wykonuje zarówno kompilację jak i linkowanie. Jeżeli podamy zbiory o rozszerzeniu .o , wykonywane jest tylko linkowanie.

Wywołanie: “g++ *.cpp”

Szybkie i proste, ale kto chce mieć program nazwany “a.out?”

Opcje kompilacji

- “-c”: kompiluje zbiory .cpp do .o ale nie linkuje.
- “-g”: generuje informacje debagera
- “-I<dir>”: dodaje katalog <dir> do listy katalogów, przeszukiwanych w trakcie dołączania plików.
- “-Wall”: zmusza g++ do wyświetlania ostrzeżeń o miejscach prawdopodobnych błędów.

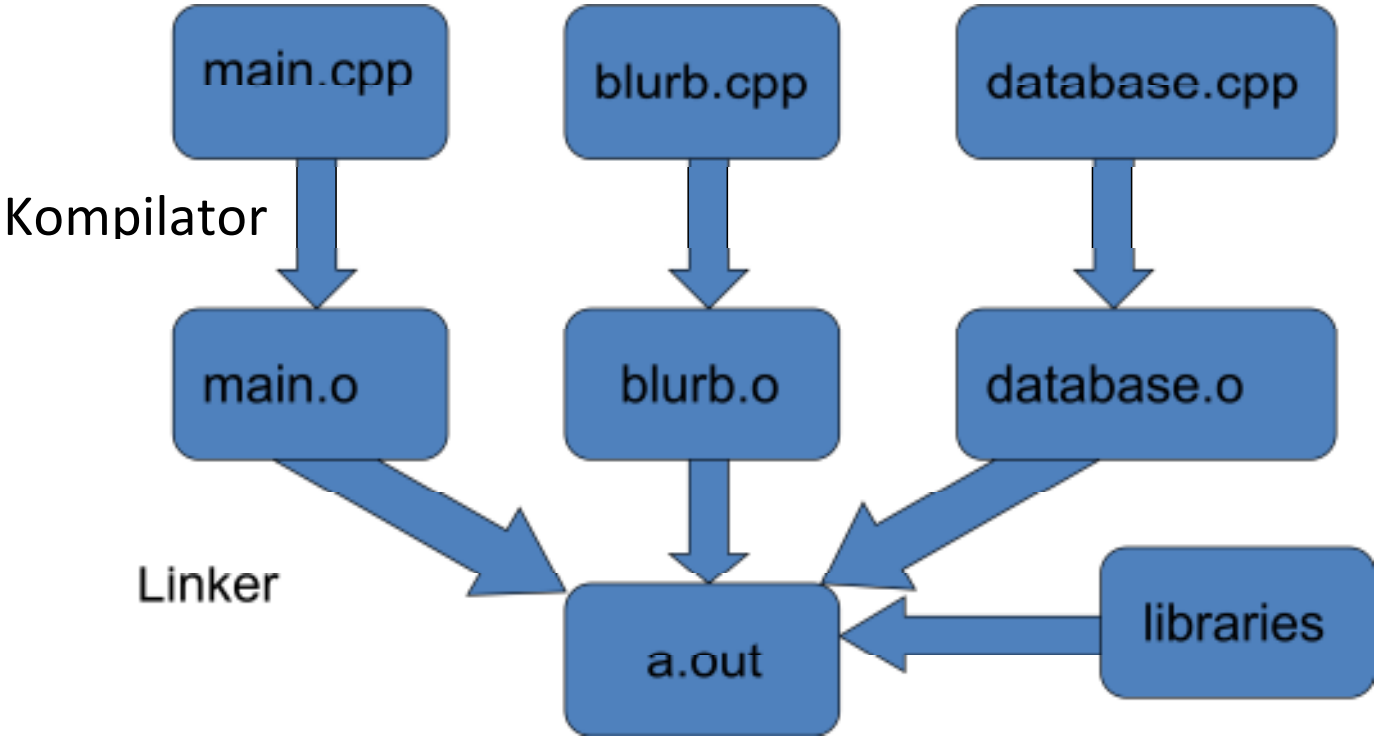
Opcje linkowania

- “-o”: określa nazwę programu, do którego wszystko razem będzie linkowane.
- “-L<dir>”: dodaje katalog <dir> do listy katalogów przeszukiwanych dla plików bibliotecznych (string, STL Vector, itp)
- “-l<libName>”: powoduje że kompilator przeszukuje bibliotekę <lib name> dla nierozwiązanych nazw podczas linkowania.

Dodatkowe informacje dotyczące kompilatorów gnu i ich opcji znajdziesz na stronie http://gcc.gnu.org/onlinedocs/.

W celu przyspieszenia, automatyzacji i odpowiedniej aktualizacji zmian operacji kompilacji oraz linkowania wielu zbiorów źródłowych wykorzystuje się zbiór Makefile z którego korzysta komenda make pod Linuxem (make uruchamiany jest w katalogu, który zawiera Makefile).

Przykładowa struktura programu o nazwie text



- Utwórz zbiór o nazwie “Makefile” który zawiera serię odpowiednio sformatowanych komend które stanowią dane wejściowe dla make.
- Taki zbiór może być duży i złożony, ale raz dobrze napisany jest wykorzystywany wielokrotnie.
- Komenda “make” wykorzystuje “Makefile” i szuka tego zbioru.

Prosty zbiór Makefile składa się z prostych “reguł” każda o postaci :

target … : prerequisites ...

command

command

command

…

- Reguła tłumaczy kiedy i jak wykonać make lub ponowny make zbioru docelowego.
- Prosty “make” wykonuje pierwszą regułę przez default. Inne reguły są uruchamiane poprzez napisanie “make <target>”

Make wymaga znaku <TAB> przed każdą komendą, bardzo często to powoduje błędy.

- “target”: to zazwyczaj nazwa kodu wykonywalnego lub zbioru typu object generowanego przez g++, ale może to być także nazwa akcji.
- “prerequisites”: lista zbiorów niezbędnych do utworzenia zbioru docelowego. Jeżeli jeden z tych zbiorów uległ zmianie to make wie że konieczne jest stworzenie zbioru docelowego ponownie. Znane także jako “dependencies” (zależności).
- “command”: akcja jaką wykonuje make, zazwyczaj kompilacja lub komenda linkowania dla g++.

Przykładowy Makefile dla zbioru docelowego text

```
text : main.o blurb.o database.o

    g++ -o text main.o blurb.o database.o

main.o : main.cpp

    g++ -c main.cpp

blurb.o : blurb.cpp blurb.h

    g++ -c blurb.cpp

database.o : database.cpp database.h

    g++ -c database.cpp

clean:

    rm -f core text main.o blurb.o database.o
```