

Języki Programowania – make

Maciej Szymański, Małgorzata Janik

1. **Make** [1] jest narzędziem służącym do automatyzacji procesu kompilacji programów składających się z wielu plików źródłowych. Jego działanie opiera się na przetwarzaniu tzw. pliku reguł: **Makefile**. Program **make** sprawdza daty ostatniej modyfikacji i na tej podstawie decyduje, które z plików źródłowych muszą być przekompilowane.

2. Podstawowym elementem pliku **Makefile** jest reguła, której składnia jest następująca:

```
CEL: SKLADNIKI
    KOMENDA
```

gdzie **CEL** to nazwa pliku wynikowego, **SKLADNIKI** to nazwy plików, na których podstawie ma być utworzony plik wynikowy, a **KOMENDA** to polecenie potrzebne do utworzenia pliku wynikowego na podstawie składników. Konieczny jest znak tabulacji przed komendą.

3. Możliwe jest definiowanie zmiennych w pliku **Makefile**. Przydatne jest użycie zmiennej zawierającej nazwy wszystkich plików będących składnikami projektu — dodanie lub usunięcie pliku z projektu będzie trzeba uwzględnić jedynie w jednym miejscu. Definicja takiej zmiennej to np.:

```
OBJS = main.o Point.o Polyline.o
```

Do zmiennej odwołujemy się przez znak dolara i nawiasy okrągłe, np.: **\$(OBJS)**.

4. Często używane są również zmienne standardowe pliku **Makefile**, np.:

- **CXX** – nazwa kompilatora języka C++
- **CXXFLAGS** – opcje kompilatora języka C++
- **LFLAGS** – opcje dla linkera

5. Przydatne są także zmienne automatyczne przechowujące wartości zmieniające się dynamicznie w trakcie przetwarzania pliku **Makefile**, np.:

- **^** – pliki składnikowe
- **@** – nazwa pliku docelowego
- **<** – aktualnie przetwarzany plik

6. Narzędzie **make** posiada tzw. reguły domyślne. Jedną z nich pozwala pominąć pisanie reguły tworzenia plików ***.o** z plików ***.cpp**. W takim przypadku **make** sam odnajduje w bieżącym katalogu pliki ***.cpp** i tworzy z nich pliki ***.o** na podstawie reguły domyślnej **\$(CXX) -c** (proszę zwrócić uwagę na komunikaty wypisywane podczas kompilacji).

7. Korzysta się również z reguły **clean** pozwalającej na usunięcie plików powstałych podczas kompilacji. Składnia tej reguły jest następująca (proszę zwrócić uwagę na pustą listę składników):

```
clean:
    rm -f *.o prog
```

8. Reguła **all** pozwala na kompilację w projektach składających się z wielu plików wynikowych, np.:

```
all: program1 program2 program3
```

9. Dobrą praktyką jest również używanie reguły **.PHONY** wraz z podaniem składników **all** i **clean**. Jej celem jest informacja dla programu **make**, iż nazwy **all** i **clean** są regułami specjalnymi, a nie nazwami plików. Przykład:

```
.PHONY: all clean
```

10. Poniżej znajduje się przykładowy plik **Makefile**, mający zastosowanie dla programu realizowanego na poprzednich laboratoriach. Komenda **make** pozwala na kompilację, natomiast komenda **make clean** usuwa pliki stworzone podczas kompilacji.

```
CXX = g++
CXXFLAGS = -Wall
LFLAGS =

OBJS = program.o student.o grupa.o

all: prog

prog: $(OBJS)
    $(CXX) $(CXXFLAGS) $^ -o $@

clean:
    rm -f *.o prog

.PHONY: all clean
```

Literatura

- [1] <http://www.gnu.org/software/make/>