10 Makefile

Głównym celem ćwiczeń jest automatyzacja procesu kompilacji:
`http://srodowisko.tcs.uj.edu.pl/files/09-power.tar.gz`.

Ćwiczenie 1. Stwórz plik `Makefile`. Napisz osobne reguły dla każdego kroku kompilacji:
- `make`, kompilacja programu na podstawie plików obiektowych,
- dla każdego pliku, jego kompilacja do pliku obiektowego,
- uwzględnij `make clean`, który skasuje pliki powstałe z kompilacji `make`.

Zapewnij, że zmiany w pliku nagłówkowym zapewnią rekompilację.

Ćwiczenie 2. Dodaj akcje `pre` i `asm`, która powodują wygenerowanie odpowiednio: kodu po preprocessingu, kodu assemblera.

Ćwiczenie 3. Kompilacja biblioteki. Dodaj dwie akcje powodujące:
- utworzenie biblioteki statycznej,
- utworzenie biblioteki dynamicznej.

Każda akcja ma zależeć od odpowiednich reguł (plików). Dodatkowo `make libs` ma kompilować obydwie wersje biblioteki, a `make clean-libs` ma czyścić powstałe pliki.

Dodaj również akcje prowadzące do kompilacji programu:
- z linkowaniem do biblioteki statycznej,
- z linkowaniem do biblioteki dynamicznej.

w razie braku bibliotek - akcja ma spowodować skompilowanie potrzebnej wersji.

Ćwiczenie 4. Struktura kodu. Zmodyfikuj `Makefile` tak, żeby:
- pliki `*.o` trafiały do `build/obj`
- program wykonywalny bezpośrednio do `build`
- skompilowane biblioteki do `build/lib`
- dodatkowo kompilacja bibliotek ma kopiować plik nagłówkowy do `build/include`

Zadbaj o odpowiednie zależności i nazwy reguł. Skoryguj również `clean` i `clean-libs`.

Ćwiczenie 5. Przepisz powyższe definiując zmienne dla nazw folderów. Zweryfikuj czy możesz wywołać `make` podstawiając pod zmienne inne lokalizacje.
Zadbaj o odpowiednie zależności w regułach.

Ćwiczenie 6. Dodaj regułę `test` która oprócz kompilacji, uruchomi program kilka razy z wybranymi stałymi argumentami. Uwzględnij fakt, że `test` nie jest plikiem i jeśli wywołamy `make test`, to reguła zawsze ma się wykonać.

Zadania

Zadanie 1. (2 punkty)
Napisz podobny `Makefile` dla programu korzystającego z `MurmurHash3.cpp`.

- struktura folderów
- kompilacja `MurmurHash` do biblioteki
- kompilacja programu

Zadanie 2. (2 punkty)
Dodaj parametry kompilacji tak aby były włączone optymalizacje i standard `C++11`. Wykorzystaj do tego zmienne, tak aby tych parametrów nie powtarzać w wielu regułach.

Wykonaj kompilację z parametrami ustawianymi w momencie wołania `make ...`