Programowanie Niskopoziomowe

Konspekt Laboratoryjny

Binutils, biblioteki statyczne i dynamiczne

Autorzy:
Gabriel Górski
Robert Gałat

29 kwietnia 2018

Spis treści

1 Informacje do zadań

W tym miejscu zakładamy, że ogólnie rozumiana teoria z seminarium jest znana uczestnikom laboratorium. Poniżej znajdują się rzeczy przydatne w wykonywaniu zadań z laboratorium.

1.1 Biblioteki statyczne

Jeśli chcemy utworzyć bibliotekę statyczną, potrzebujemy pliki obiektowe które mają się w niej znaleźć. Komenda wygląda następująco

```
ar rcs libNAZWA.a plik1.o plik2.o plik3.o
```

Żeby wykorzystać tą bibliotekę należy ją **zlinkować** z resztą plików obiektowych (na tym etapie symbol *musi* się zawierać w którymś z nich).

1.2 Biblioteki współdzielone

Podobnie ma się z bibliotekami dynamiczną, tutaj jednak należy pamiętać o większej ilości niuansów.

Po pierwsze składowe pliki obiektowe biblioteki należy skompilować z flagą **fPIC**.

Tworzenie biblioteki:

```
gcc -shared plik1.o plik2.o plik3.o -o libNAZWA.so
```

Wykorzystanie biblioteki:

```
gcc -L . plik.o -o plik -lNAZWA
```

2 Zadania

2.1 Biblioteki statyczne

- Celem zadania jest uzupełnienie pliku **run.sh** tak, aby umożliwał on kompilację biblioteki statycznej, oraz zlinkowanie projektu do programu wynikowego. [**1a**]

- Celem zadania jest uzupełnienie pliku **run.sh** tak, aby stworzyć biblioteki statyczne oraz zlinkować je z *głównym* plikiem obiektowym. Czy zauważasz coś ciekawego? Jeśli tak, to czy potrafisz to wyjaśnić? [**1b**]

2.2 Biblioteki współdzielone

- W tym zadaniu należy utworzyć bibliotekę dynamiczną, zlinkować wobec niej plik obiektowy, a następnie otrzymany plik wykonywalny należy uruchomić — pamiętaj o odpowiednich flagach kompilacji i linkowania! [**2**]
- Celem zadania jest podmienienie implementacji funkcji która była w bibliotece z poprzedniego zadania.

  Należy to zrobić bez bez modyfikacji pliku wykonywalnego z poprzedniego zadania tj. poprzez wykorzystanie funkcjonalności linkera dynamicznego.

  Wprowadź własną implementację tej funkcji. [**3**]

2.3 Binutils

- W tym zadaniu należy dokonać kompilacji pliku *relocatableFile.c* a następnie przeanalizować wygenerowany plik binarny programem *nm* oraz *objdump* [**4**]

2.4 Pluginy i dynamiczne ładowanie

- Celem zadania jest uzupełnienie pliku *main.c* w taki sposób aby uruchomić funkcję z biblioteki libgoo.so, która powinna zostać załadowana w czasie działania programu. [**5**]
- Celem zadania jest uzupełnienie brakujących części obsługi pluginów, oraz napisanie własnego pluginu, wzorując się na przygotowanym przykładzie

  Do uzupełnienia są następujące funkcje:[**6**]

  - apply_hook() {PluginManager/PluginManager.c}
  - initPlugi() {PluginManager/PluginLoader.c}