Wydział Fizyki i Informatyki Stosowanej

Programowanie Niskopoziomowe

Konspekt

Aleksandra Poręba
nr. indeksu 290514

30 marca 2019

Spis treści

1 Abstrakt

Projekt jest podzielony na dwie części: o bibliotekach statycznych oraz o wtyczkach (pluginach). Całość jest realizowana w języku C.

2 Tworzenie bibliotek statycznych

2.1 Wymagania

2.1.1 Program ar

Do tworzenia bibliotek statycznych może zostać użyty program `archiver`. Aby stworzyć bibliotekę należy użyć opcji `r` oraz `c`, służących odpowiednio do dodania plików do archiwum oraz stworzenia archiwum.

```
ar rc [nazwa biblioteki] [pliki obiektowe]
```

Nazwa powinna zaczynać się od `lib`, a kończyć rozszerzeniem `.a`. Dodatkową, pomocną komendą może być:

```
nm [nazwa_biblioteki.a]
```

wypisująca zawartość biblioteki wraz z eksportowanymi przez pliki symbolami.

2.1.2 Kompilacja z biblioteką statyczną

Aby skompilować plik za pomocą `gcc` z załączonymi statycznie bibliotekami należy dodać następujące flagi:

```
-L[ścieżka do bibliotek]
-l[nazwa biblioteki]
```

2.1.3 Dodatkowe

Podczas wykonywania zadania należy pamiętać o kolejoności załączania bibliotek. Problem został przedstawiomy na seminarium.

2.2 Treść zadania

Zadanie polega na dopisaniu brakującego pliku biblioteki, a potem stworzenie jej, by można było skompilować program za pomocą komendy

```
gcc main.c -L./lib -lfun2 -lfun1
```

bez żadnych errorów/warningów. Plików `main.c` oraz `fun1.c` nie można zmieniać!

3 Tworzenie systemów wtyczek

3.1 Wymagania

3.1.1 Koncepcja systemów wtyczek

Wtyczki są tworzone jak biblioteki dynamiczne. Program powinien przeszukiwać wskazany folder z potencjalnymi wtyczkami, i jeżeli owe znajdzie załączać je do programu. W przypaku tego zadania, przyjmiemy uproszczoną wersję, pluginy będą zapisane w tablicy, podane ź góry". Każda wtyczka powinna udostępniać funkcje zgodne z zaprojektowanym interfejsem, które będą wywoływane przez program główny.

3.1.2 Biblioteka dlfcn

Do obsługi wtyczek można użyć biblioteki `dlfcn`.

- `dlopen` - używamy do otwarcia pluginu, jako argument podajemy nazwę pliku wraz ze ścieżką oraz `RTLD_NOW` oznaczającym rozwiązanie wszystkich niezdefiniowanych symboli,
  np. `dlopen("./plugins/p1.so", RTLD_NOW);`
- `dlsym` - pobiera wskaźnik do funkcji o podanej nazwie, jako argumenty przyjmuje "handler"do biblioteki, zwrócony przez `dlopen` oraz nazwę funkcji
  np `dlsym(handle, "process");`
- `dlclose` - zamyka "handler"do biblioteki

3.1.3 Dodatkowe

3.2 Treść zadania

Zadaniem jest napisanie prostego systemu wtyczek. Zakładamy że nazwy wtyczek podane są w tablicy `plugins`, i znajdują się w folderze o tej samej nazwie. Program powinien przyjmować od użytkownika ciąg znaków (jako argument wywołania lub przez sprintf). Wtyczki powinny być dwie i realizować różne działania na owym stringu, a następnie wypisywać je na ekran. Przykładowe działania to: zamiana wszystkich liter na wielkie, mieszanie kolejność liter, wypisywanie tylko samogłosek, wypisywanie wyrazu od tyłu etc.

Należy pamiętać o uzupełnieniu `makefile`! Wskazówki w którym miejscu dokonać modyfikacji znajdują sie bezpośrednio w przygotowanym pliku.

4 Tworzenie wtyczek

4.1 Wymagania

4.1.1 Program bazowy i jego interfejs

Program, do którego zadaniem będzie napisać wtyczki, został omówiony dokładnie na seminarium. Do rozwiązania tego zadania, musimy jedynie znać interfejs, jaki ma mieć plugin. Składa się na niego funkcja inicjalizująca (rejestrująca), funkcja wypisująca opcję pluginu jako opcja w menu oraz funkcja realizująca działanie.

4.1.2 Dodatkowe

4.2 Treść zadania

Celem zadania jest napisanie wtyczki do programu kalkulatora prezentowanego na zajęciach. Jej budowa musi być zgodna z zaprojektowanym interfejsem. Wtyczka powinna dodać obsługę mnożenia/dzielenia/potęgowania.
Należy pamiętać o uwzględnieniu nowej wtyczki w `makefile`!