CourseWare Wiki

NAVIGATION

B4B35OSY

Cvičení

material

Cvičení 1 : Úvod do UNIXových OS

Cvičení 2 : Skriptovací jazyk BASH

Cvičení 3: Zpracování textu v shellu a regulární výrazy

Cvičení 4 : Vytváření procesů v C a překlad make

Cvičení 5 : Vlákna a synchronizace

Cvičení 6 : Další metody synchronizace

Cvičení 8 : Systémová volání a inline assembler

Cvičení 9 : Útok pomocí přetečení zásobníku

Cvičení 10 : NOVA – Systémová volání a správa paměti

Cvičení 11 : Uživatelský paměťový alokátor pro OS NOVA

Cvičení 12 : Podpora více vláken v OS NOVA

Cvičení 13 : Sestavení Linuxového systému

MY COURSES

WINTER 2019 / 2020

B4B33ALG – Algoritmizace

B4B35OSY – Operační systémy

B6B36OMO – Objektový návrh a modelování

SUMMER 2018 / 2019

–Table of Contents

Cvičení 3: Zpracování textu v shellu a regulární výrazy

Domácí příprava

Nastudujte použití regulárních výrazů a nástroje pro zpracování textu (alespoň zběžně **grep**, **sed** a **tr**):

Zadání úlohy

Vytvořte skript v jazyce BASH, podle následujících požadavků:

- Skript bude mít příponu `.sh` .
- Skript bude akceptovat parametry podle následujícího vzoru `skript.sh [OPTIONS] [FILE…]`
- Pokud není uveden žádný `FILE` , skript bude číst data ze standardního vstupu a výstup bude vypisovat na standardní výstup. V textu níže označuje slovo `FILE` i data ze standardního vstupu.

- U všech souborů `FILE` (může jich být více) předpokládá, že to jsou zdrojové soubory v jazyce C a provede na nich operace podle zadaných přepínačů.
- Bude mít 4 přepínače určující funkci skriptu:
  - `-h` vypíše stručnou nápovědu a ukončí skript. Kdykoliv je zadán, ostatní následující přepínače jsou ignorovány.
  - `-i <prefix>` provede změnu názvů souborů v direktivě `#include` ve všech souborech `FILE` tak, že `<prefix>` přidá na začátek názvu souboru. Uvažujte obě varianty direktivy `#include` – jak pro soubory z aktuálního adresáře (`"foo.h"`) tak pro systémové knihovny (`<foo.h>`). Například pro `-i lib/` , bude `#include "foo.h"` změněno na `#include "lib/foo.h"` .
  - `-r` změní konvenci názvů funkcí použitých v souborech FILE (definice, deklarace a jejich volání). Pokud je název funkce tvořen slovy oddělenými jedním podtržítkem, odstraní podtržítka a pokud je následující znak písmeno změní ho z malého na velké. Jako slovo je považována kombinace malých písmen a číslic o jednom či více znacích. Například `toto_je_funkce_2_priklad()` se změní na `totoJeFunkce2Priklad()` . Nebude se měnit: `_func()` , `_func_xy()` , `sin_Cos()` , `remove_allX()` a `internal__func()` . Předpokládejte, že mezi názvem funkce a levou závorkou není žádný komentář ani konec řádku.
  - (nepovinné, 1 bod) `-f <regex>` v kombinaci s `-r` omezí refaktoring jen na funkce, jejichž jméno odpovídá regulárnímu výrazu `<regex>` . (priklad: `uloha2.sh -r -f '^my' *.c` změní pouze názvy funkcí začínajících "my")
  - při zadání přepínačů `-i` i `-r` skript vykoná obě funkce, bez těchto přepínačů skript soubor nezmění
  - při zadání jiného přepínače skript vypíše nápovědu a skončí s návratovou hodnotou 1
- Všechny změny se provedou i uvnitř komentáře, nebo řetězce.
- Předpokládejte, že zdrojové kódy je možné zkompilovat.
- Není-li řečeno jinak, skript končí s návratovou hodnotou 0.

Poznámky k implementaci

- Pro řešení úlohy doporučujeme použít programy grep, sed a tr
- Jak je definován identifikátor v C se můžete dočíst zde 'https://www.studytonight.com/c/keywords-and-identifier.php'
- Změnu z malých písmen na velká provedete například příkazem `sed 's/./\U&/'` (viz dokumentaci), nebo `tr '[:lower:]' '[:upper:]'` .
- Standardní chybový výstup můžete použít pro výpis ladících hlášek (`echo >&2 …` nebo `set`

```
-x )
```

Materiály

Regulární výrazy

Domácí příprava na další cvičení

Předpokládáme, že máte základní znalosti jazyka C a víte, jak funguje překlad ze zdrojových kódů jazyka C do binární spustitelné aplikace (v obecném případě, kdy je zdrojových souborů více).

Dále byste měli mít alespoň minimální povědomí o použití překladače **gcc** a jeho základních parametrech (gcc)

Nastudujte si použití nástroje **make** pro překlad programu v jazyku C/C++: make

Dále je pro absolvování cvičení nutné mít přehled o systémových voláních fork, pipe, dup, open, kill, wait a exec, tzn. měli byste vědět jak vzniká nový proces a jak lze přesměrovat standardní vstup a výstup. Potřebné informace se dozvíte na některé z předchozích přednášek.

courses/b4b35osy/cviceni/cviceni03_text.txt · Last modified: 2019/10/04 13:36 by sojkam1