Implementační dokumentace k projektu do IPP 2017/2018
Jméno a příjmení: Filip Kočica
Login: xkocic01

1 Úvod

Jak parse.php, tak interpret.py jsem implementoval hned na začátku školního roku, abych měl později dost času na ostatní projekty, což se ukázalo jako užitečné, ale bohužel teď, při psaní dokumentace, si již nepamatuji přesný postup implementace.

2 Skript parse.php

Tento skript měl za úkol provést lexikální a syntaktickou analýzu vstupního kódu v `IPPcode18`, k tomu jsem využil zkušeností s konečným automatem nabytých v kurzu `IFJ` a po jeho návrhu jsem jej implementoval a otestoval jeho schopnost rozeznat jednotlivé lexémy a nalézt možné chyby.

Tokeny se postupně předávají do parseru, jež provádí syntaktickou analýzu. Po zjištění o jakou instrukci se jedná, na základě vědomosti o tom co následuje této instrukci volám funce `symbolCheck` a `variableCheck`, jež provedou kontrolu symbolu – může být proměnná i konstanta a kontrolu proměnné samotné.

3 Skript interpret.py

Při implementaci tohoto skriptu jsme měli možnost použít několik knihoven. K rozparsování argumentů jsem tedy využil `argparse`, jež podle mě přináší více trápení, než užitku (oproti jednoduchému a přímočarému optarg) a pro rozparsování vstupního XML jsem použil knihovu `XMLElementTree`.

Poté už je implementace založena na postupném procházení jednotlivých elementů XML a jejich pod–elementů, kontrolami a prováděním akcí na základě typu instrukce.

Problém se vyskytl při prázdných řetězcích, kdy přístup na tento prvek vyhodil vyjímku. Proto bylo potřeba na začátku projít celé XML a všechny prvky s hodnotou `None` nastavit na prázdný řetězec. Při tom jsem si také uložil jména jednotlivých návěští, abych při případném skoku na toto návěští mohl provést kontrolu, zda opravdu existuje.

Největším problémem bylo se vyznat v a vracet správné chybové kódy. Po vytvoření sady testů otestované pomocí testovacího rámce `testp.php` se mi podařilo víceméně vždy vracet správné chybové kódy specifikované v zadání a na fóru.

4 Skript test.php

Skript iterativně prochází zadaný nebo aktuální adresář a jeho podadresáře a hledá soubory s příponou `.src`. Poté zkusí najít i ostatní soubory se jménem stejným jako soubor s příponou `.src` a pokud neexistují, vytvoří je prázdné.

Tato úloha bylo implementována pomocí funkce `shell_exec`, jež spouští příkazy v linuxovém terminálu a my tedy můžeme spouštět skripty `parse` a `interpret` a přesměrovávat jejich vstupy/výstupy. K implementaci jsem použil linuxové utility diff, cat a echo.